Introduction to Zero-Knowledge Proofs

Thibauld Feneuil

Winter Research School

February 20, 2024 — Rennes (France)

CryptoExperts
WE INNOVATE TO SECURE YOUR BUSINESS

Speaker

- Thibauld Feneuil
- Cryptographer @ *CryptoExperts*
- *PhD Defense in October 2023*
- *Main Research Area: MPC-in-the-Head paradigm,* post-quantum signatures

Speaker

- Thibauld Feneuil
- Cryptographer @ *CryptoExperts*
- *PhD Defense in October 2023*
- *Main Research Area: MPC-in-the-Head paradigm,* post-quantum signatures

Lectures

- 9h-10h30: Introduction to Zero-Knowledge Proofs
- 11h-12h30: Post-Quantum Signatures from Secure Multiparty Computation

Modern Cryptography

- Secure Communication
  - Encryption for the confidentiality
  - MAC / signatures for the authentication and integrity

Modern Cryptography

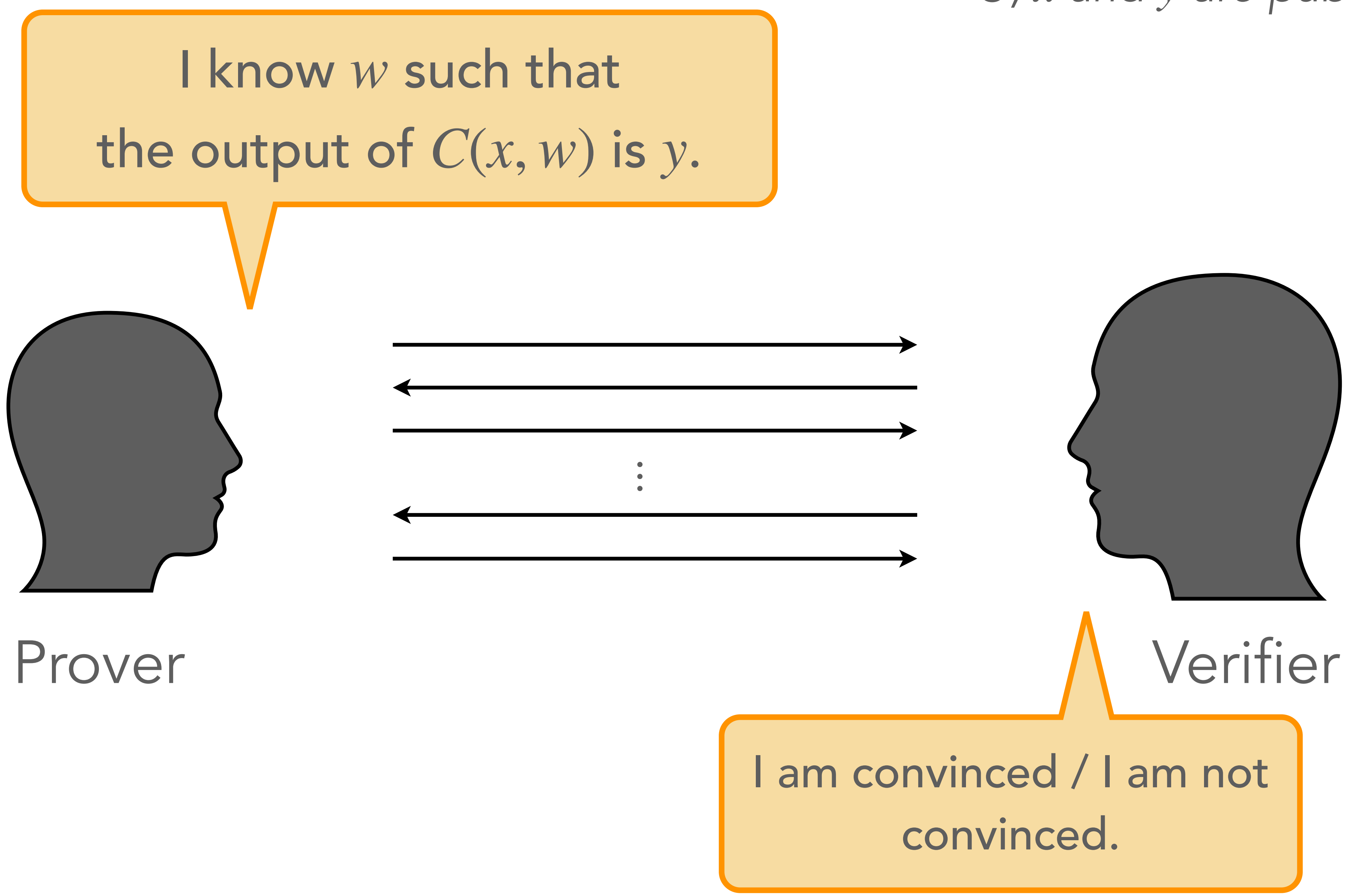
- Secure Communication
  - Encryption for the confidentiality
  - MAC / signatures for the authentication and integrity
- Secure Computation
  - Multiparty Computation
  - Homomorphic Encryption
  - (Zero-Knowledge) Proof Systems
  - …

Modern Cryptography

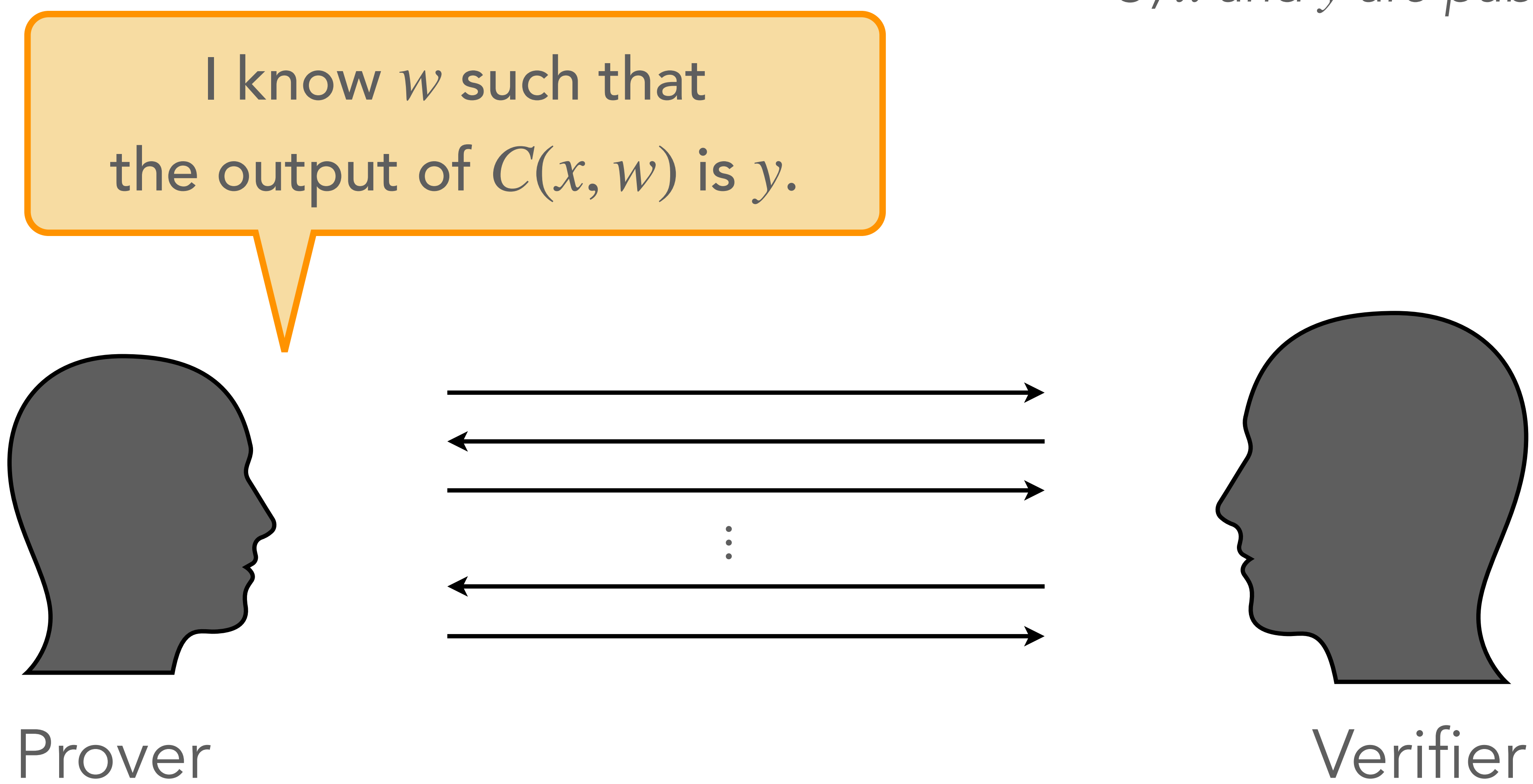
- Secure Communication
  - Encryption for the confidentiality
  - MAC / signatures for the authentication and integrity
- Secure Computation
  - Multiparty Computation
  - Homomorphic Encryption
  - **(Zero-Knowledge) Proof Systems**
  - …

Definition and Properties

C, x and y are public

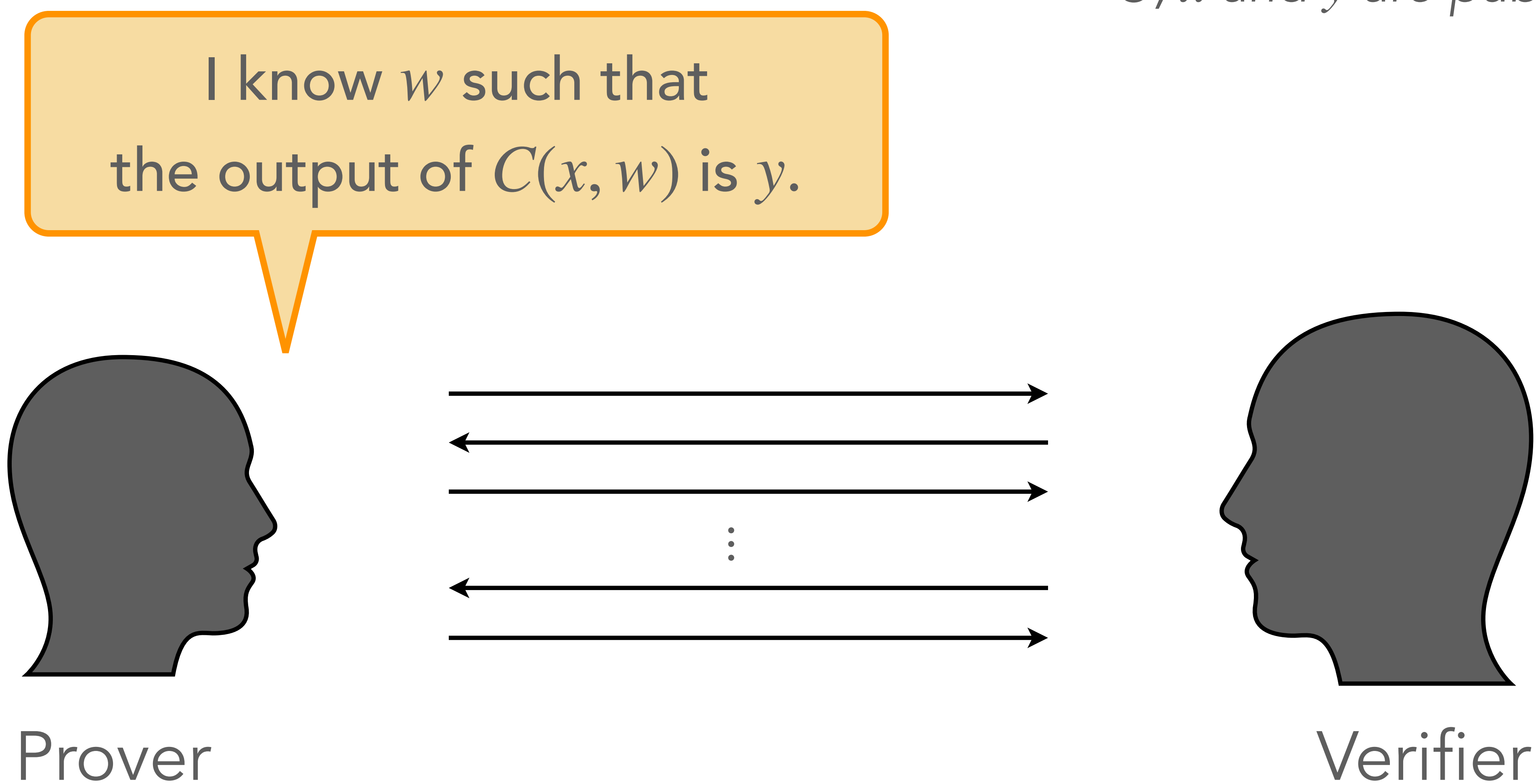


C, x and y are public



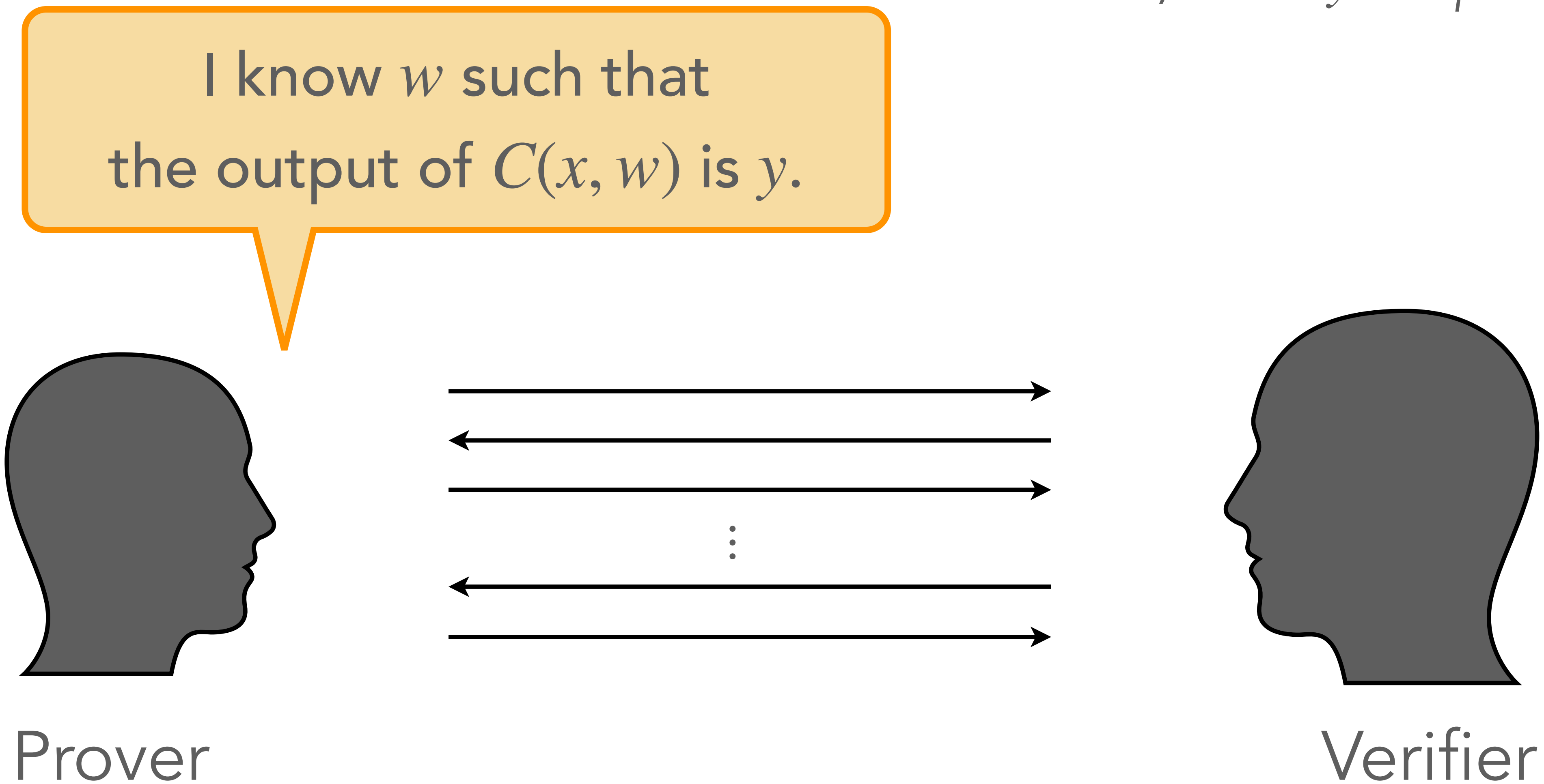
- I know two non-trivial integers p and q such that $p \cdot q = 15$.
    - x is not used
    - w is the couple (p, q)
    - C is the multiplication algorithm
    - y is the value 15

C, x and y are public



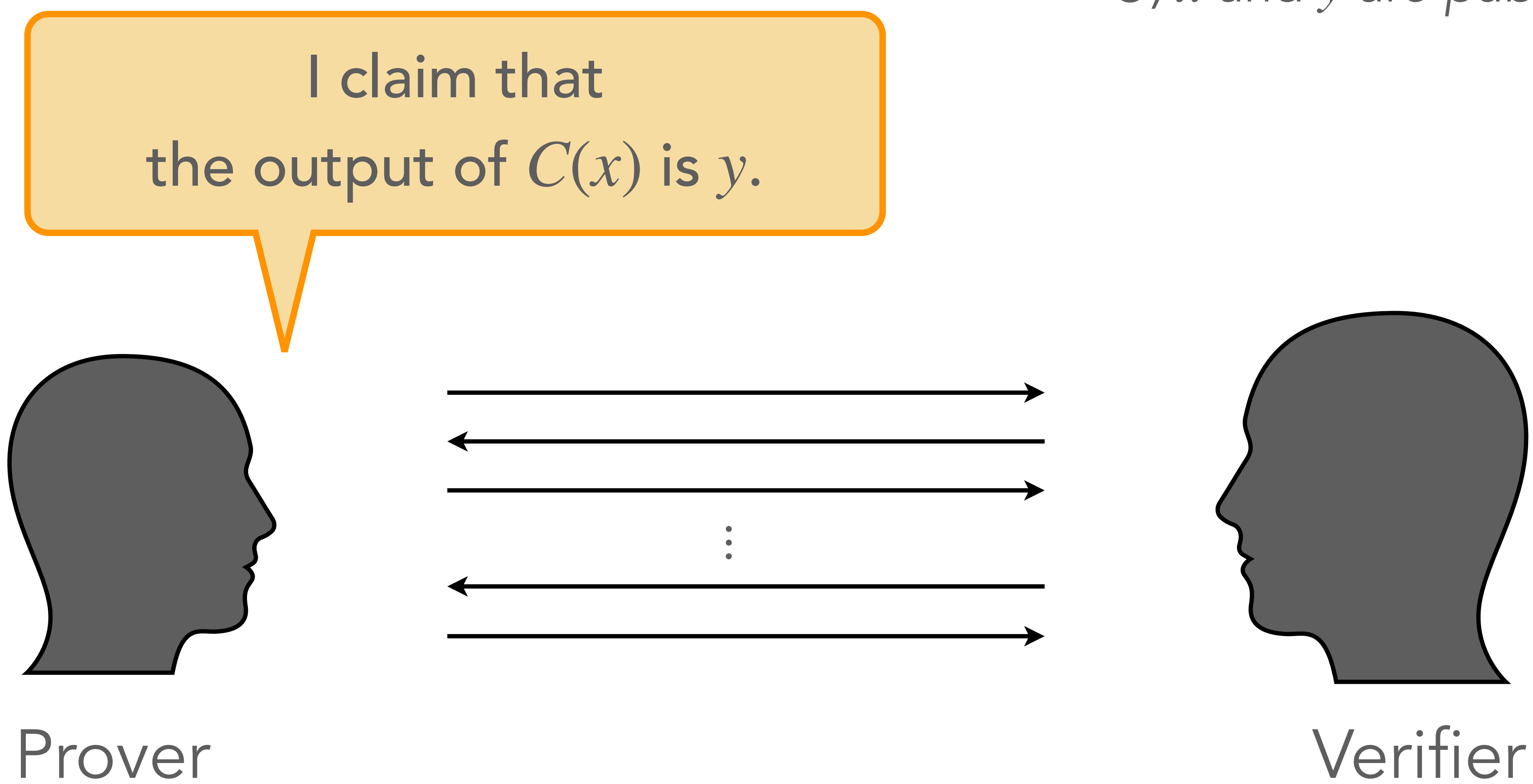
- Given a graph G, I know a 3-coloration of this graph.
  - x is the graph G
  - w is the 3-coloration
  - C is the verification algorithm
  - y is the value “True”

C, x and y are public



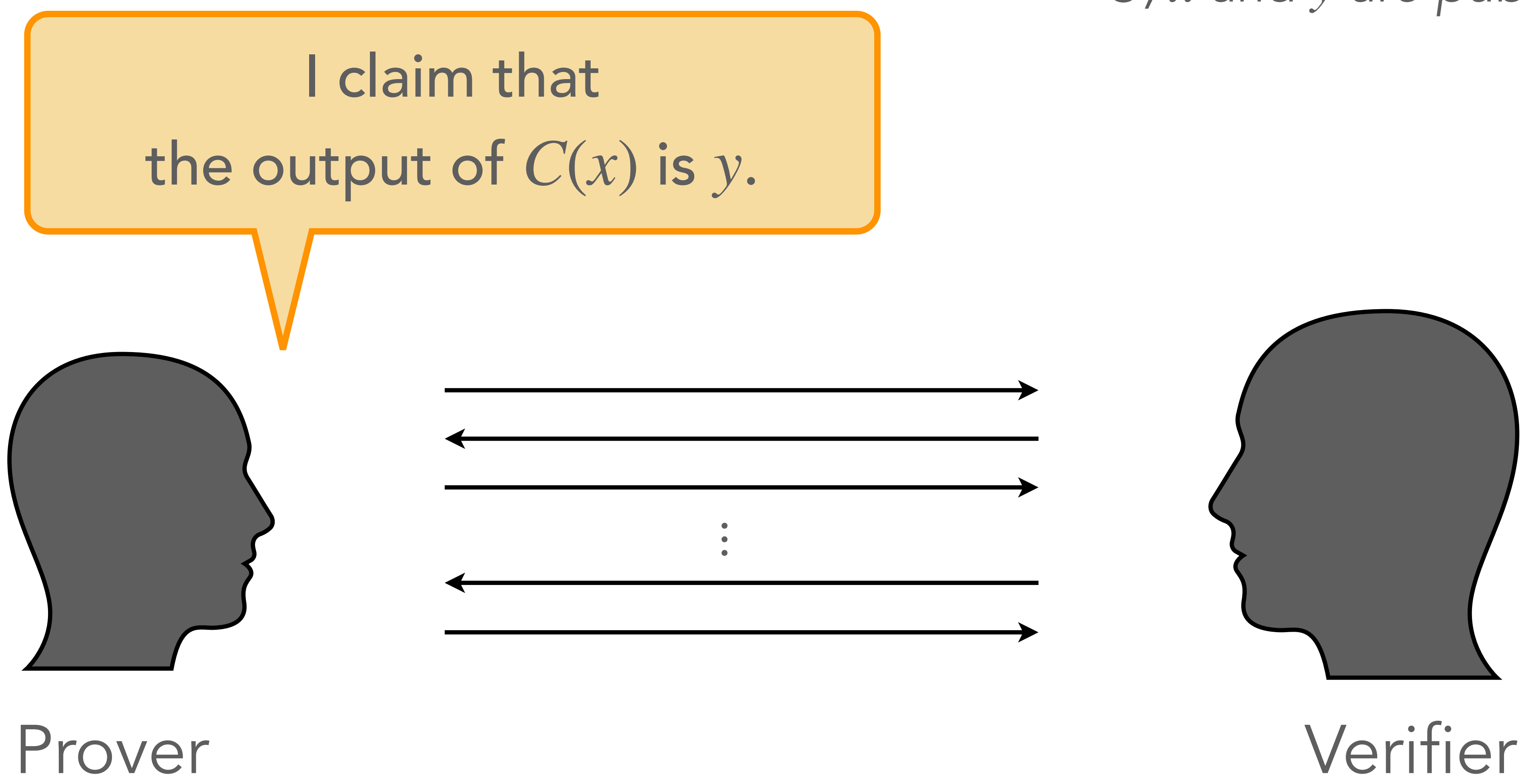
- I know the password of this encrypted file.
  - x is the encrypted file
  - w is the password
  - C is an algorithm checking if the password is valid
  - y is the value "True"

C, x and y are public



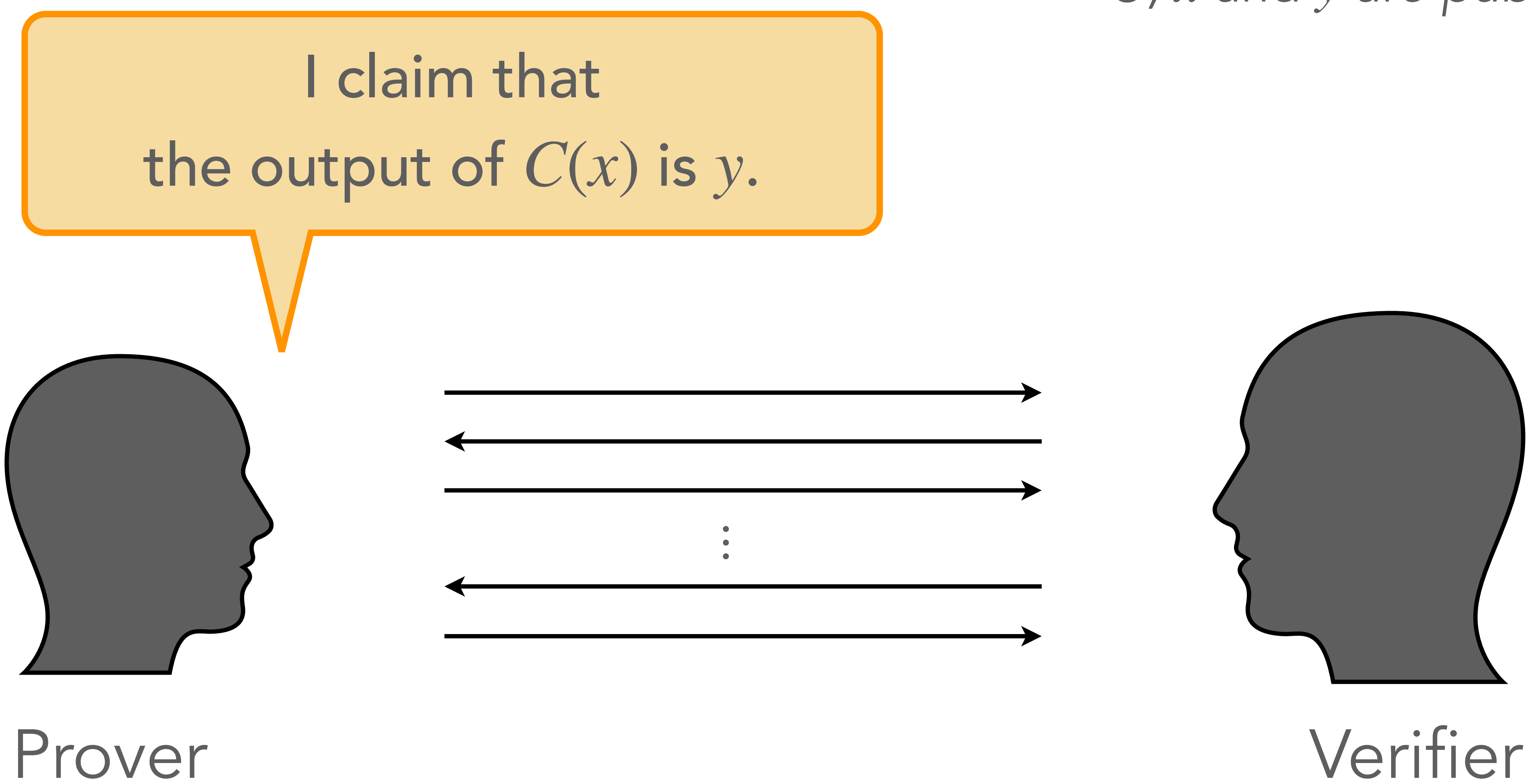
- I claim that the 100th term of the Fibonacci sequence is 354224848179261915075.
    - x is the index 100
    - C is the algorithm for the Fibonacci sequence
    - y is the integer 354224848179261915075.

C, x and y are public

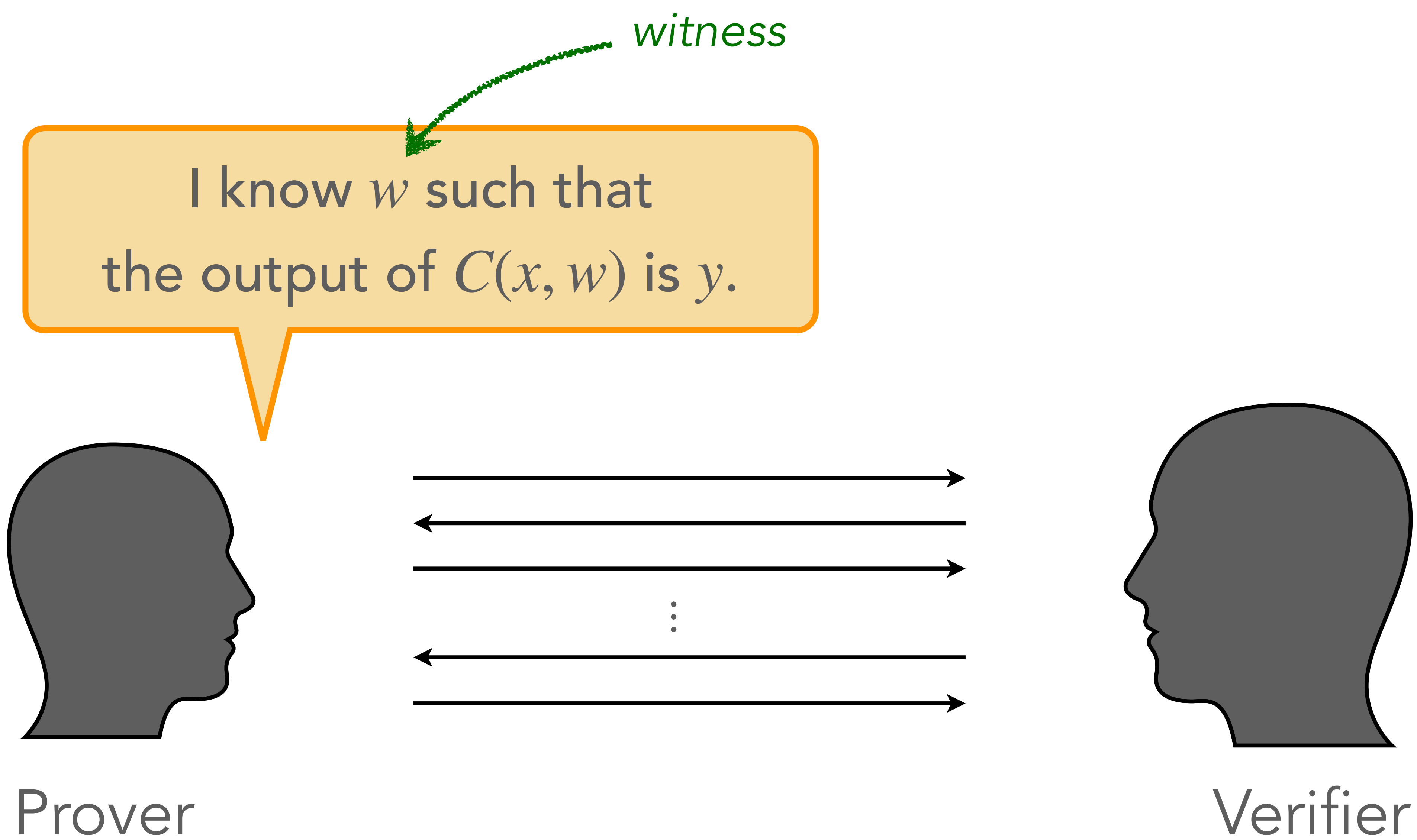


- I claim that the product of the two matrices X and Y is equal to the matrix Z.
    - x is the two matrices X and Y
    - C is the multiplication algorithm
    - y is the matrix Z

C, x and y are public

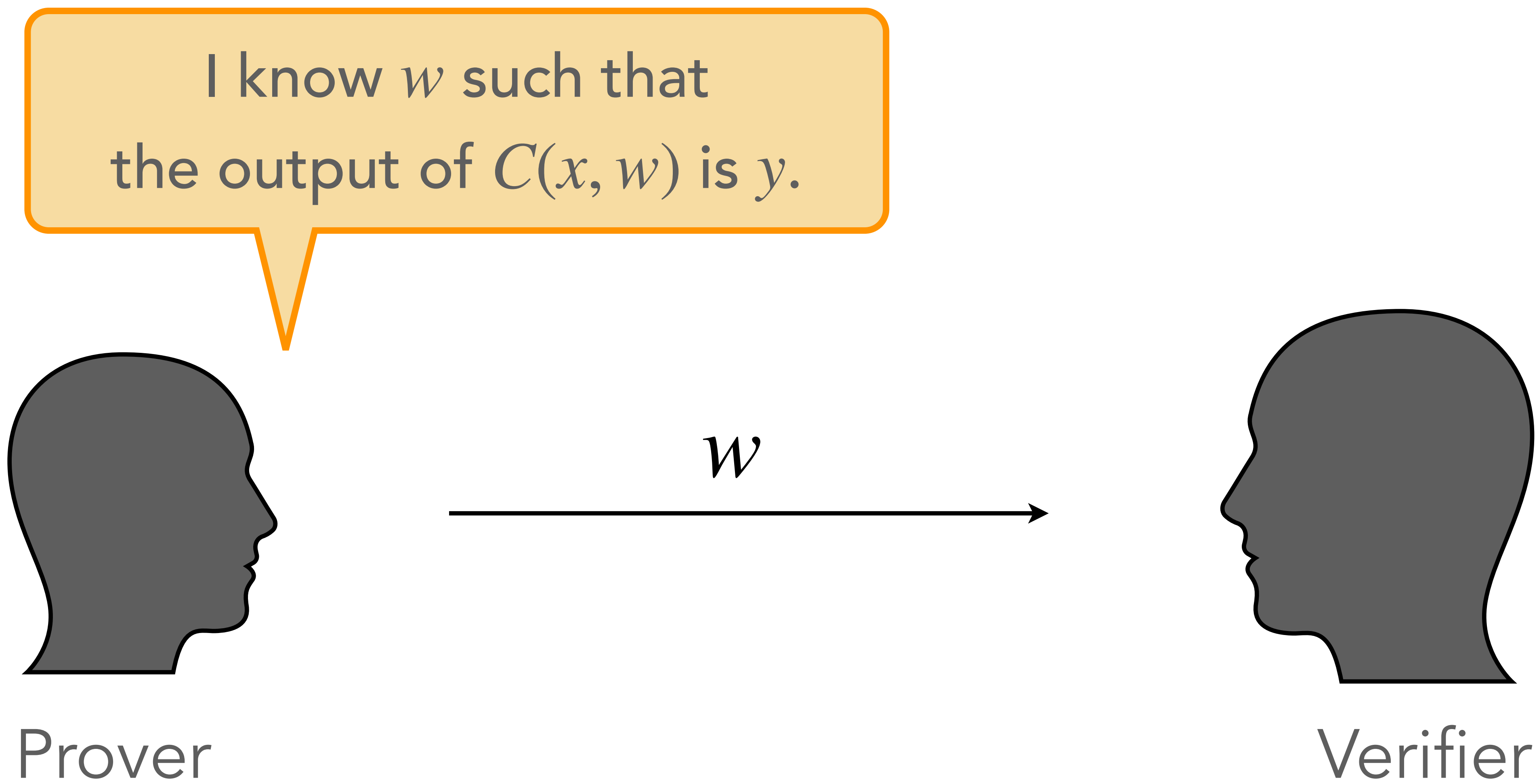


- I claim that the number of occurrences of the word “mais” in the book “A la Recherche du Temps Perdu” written by Marcel Proust is 8256.
  - x is the text of the book
  - C is the counting algorithm for the word “mais”.
  - y is the number 8256



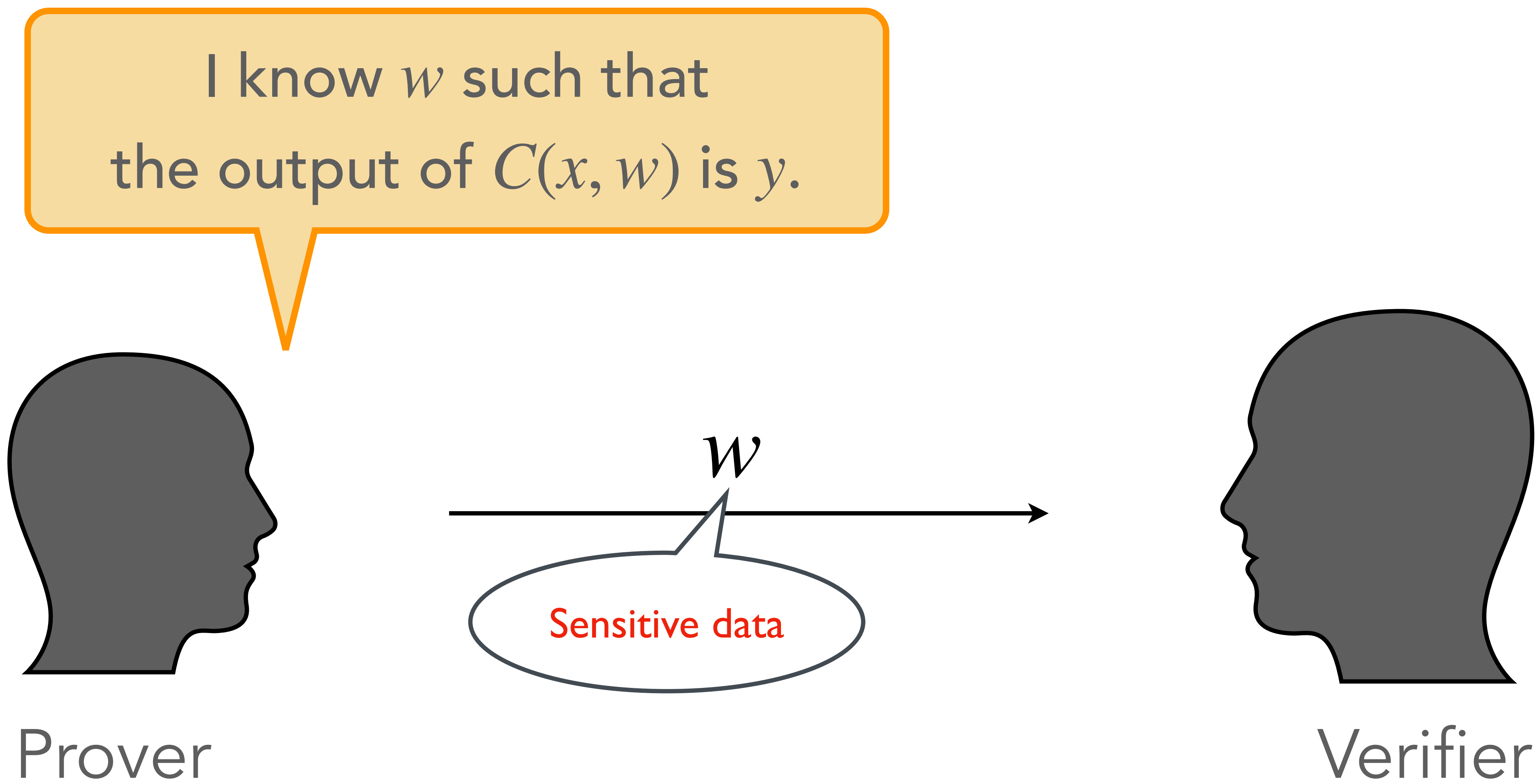
- Completeness: if the prover is ***honest*** (*i.e.* if his claim is correct), the verifier should be ***convinced*** at the end of the discussion.

- Soundness: if the prover is ***malicious*** (*i.e.* if his claim is invalid), the verifier should ***not*** be ***convinced*** at the end of the discussion.

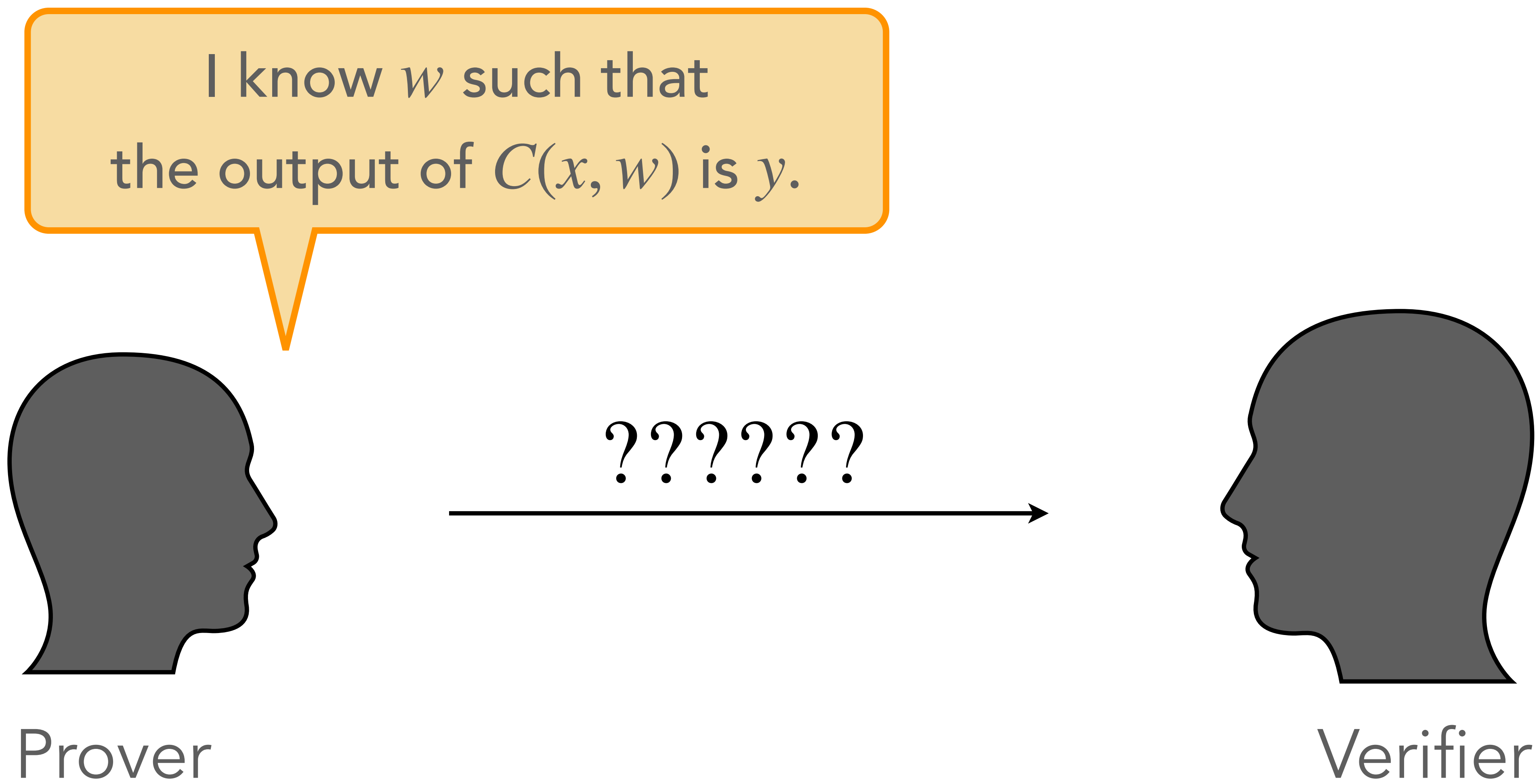


- Completeness: if the prover is ***honest*** (*i.e.* if his claim is correct), the verifier should be ***convinced*** at the end of the discussion.

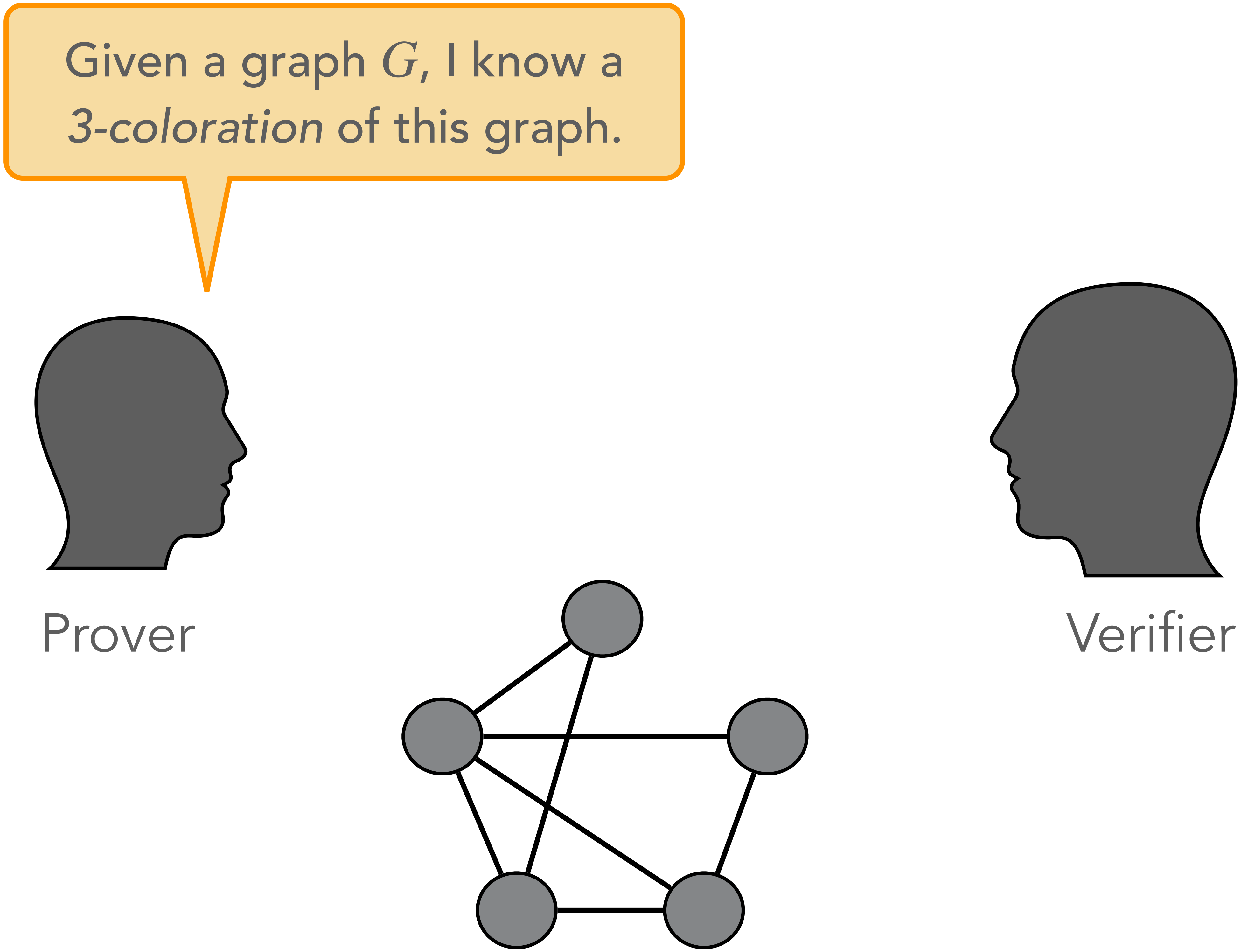
- Soundness: if the prover is ***malicious*** (*i.e.* if his claim is invalid), the verifier should ***not*** be ***convinced*** at the end of the discussion.

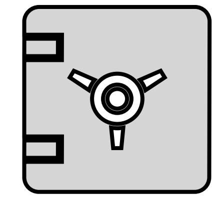


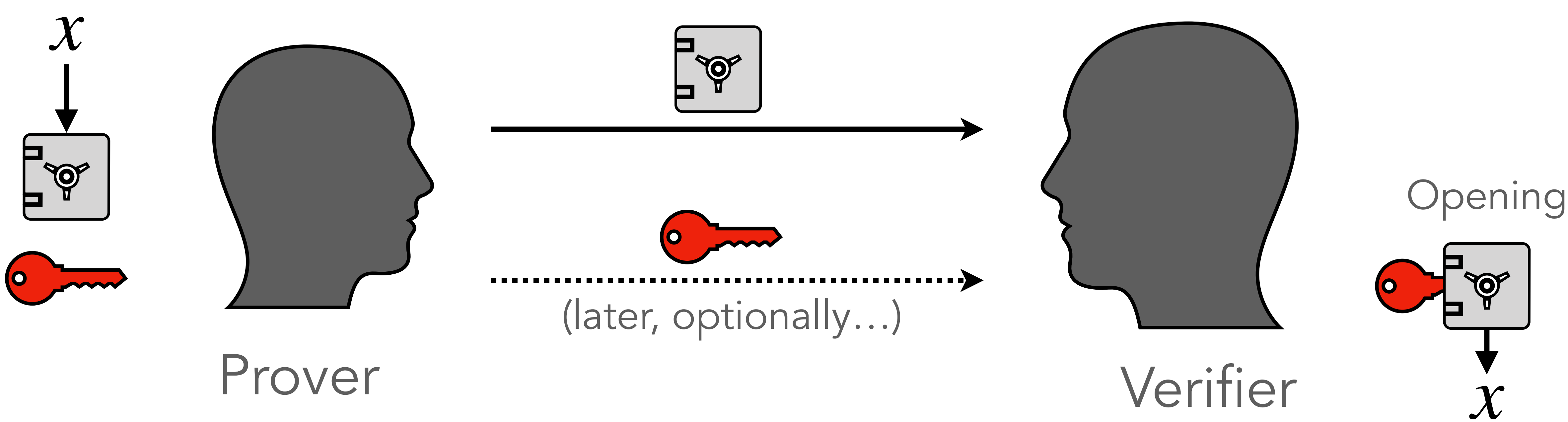
Zero-Knowledge: the verifier should ***learn nothing*** about the witness w, not even a partial information.

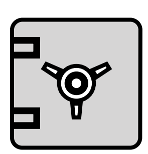
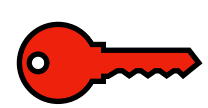


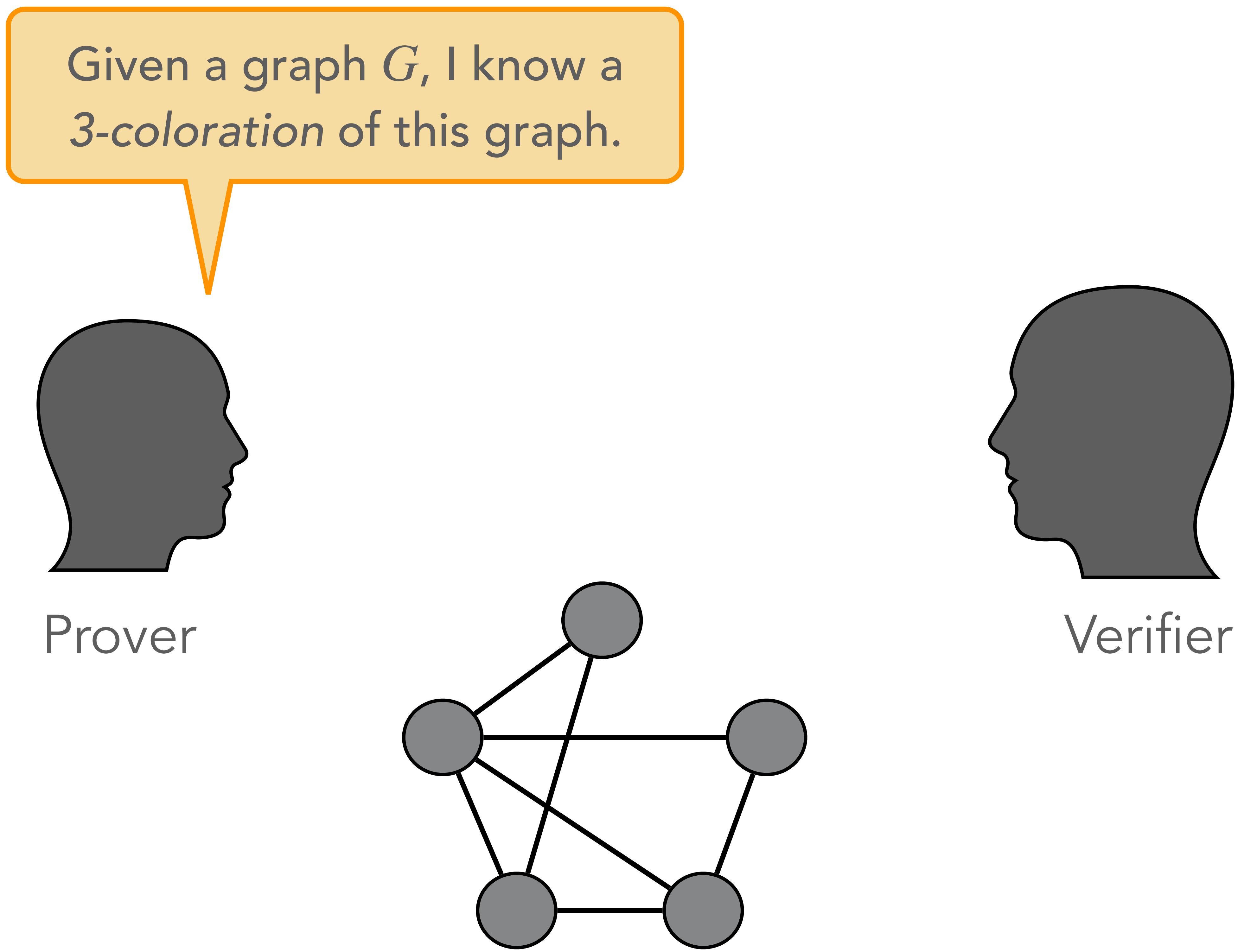
Zero-Knowledge: the verifier should ***learn nothing*** about the witness w, not even a partial information.

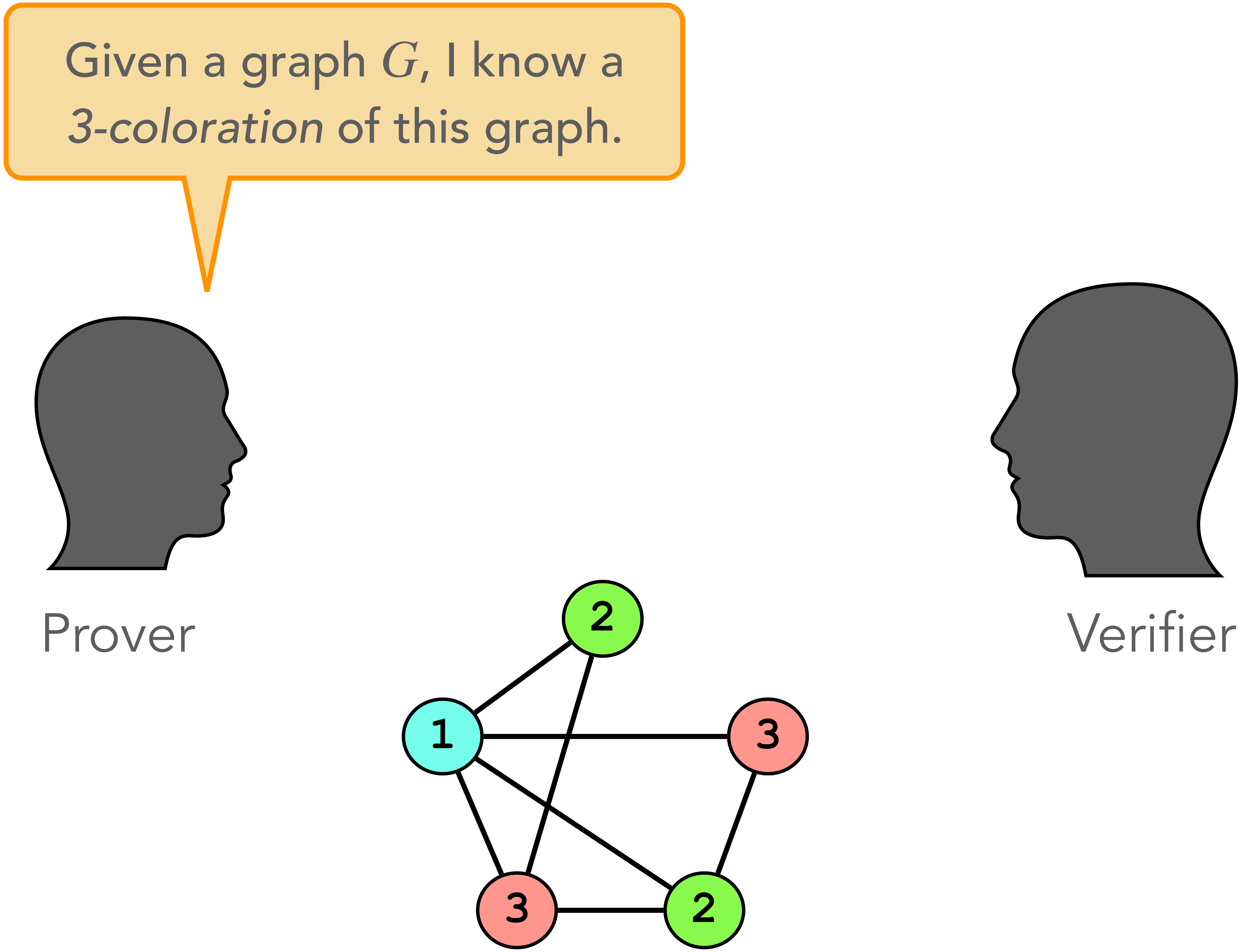

Given a graph G, I know a
3-coloration of this graph.
Prover
Verifier

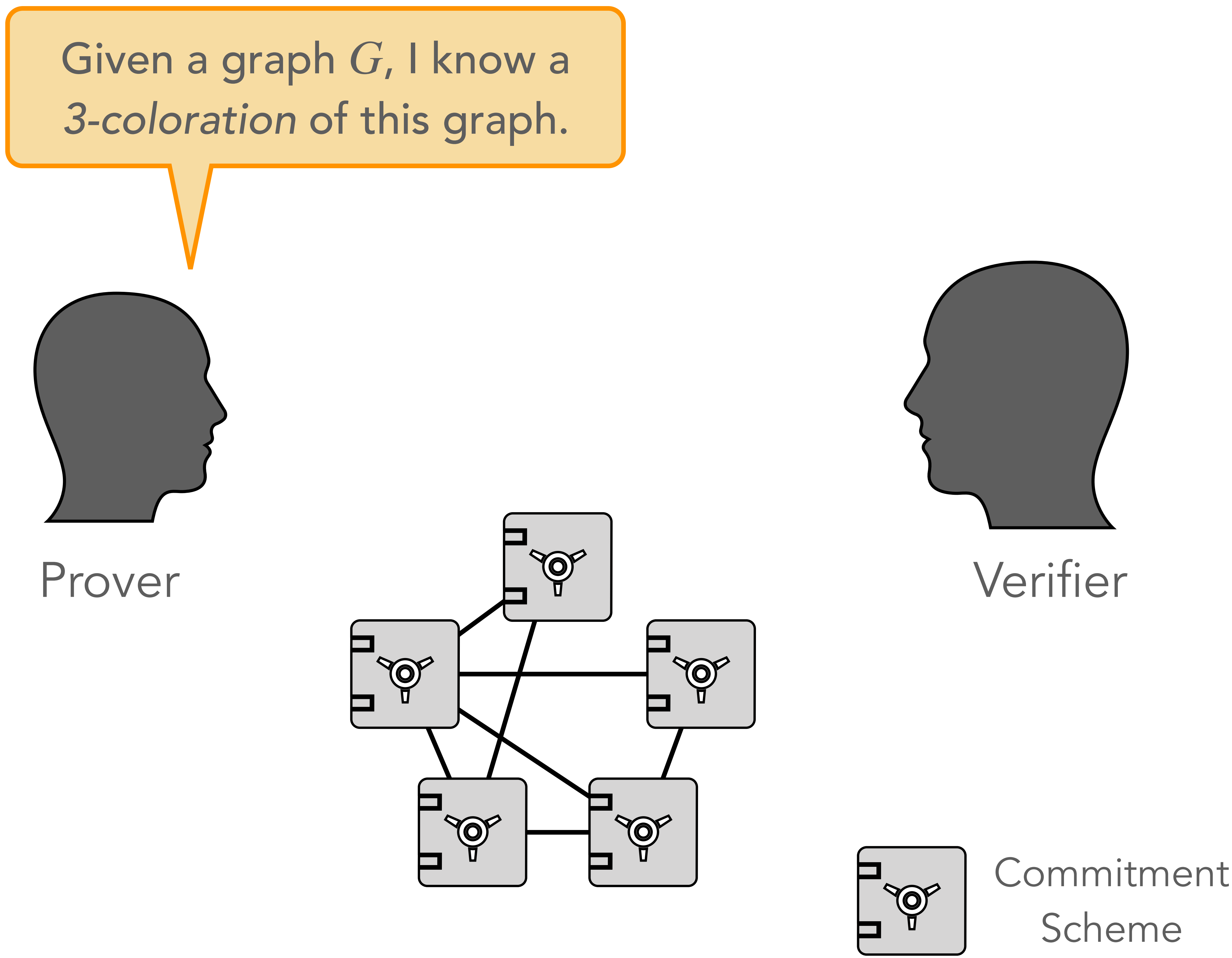
Background: Commitment Scheme 

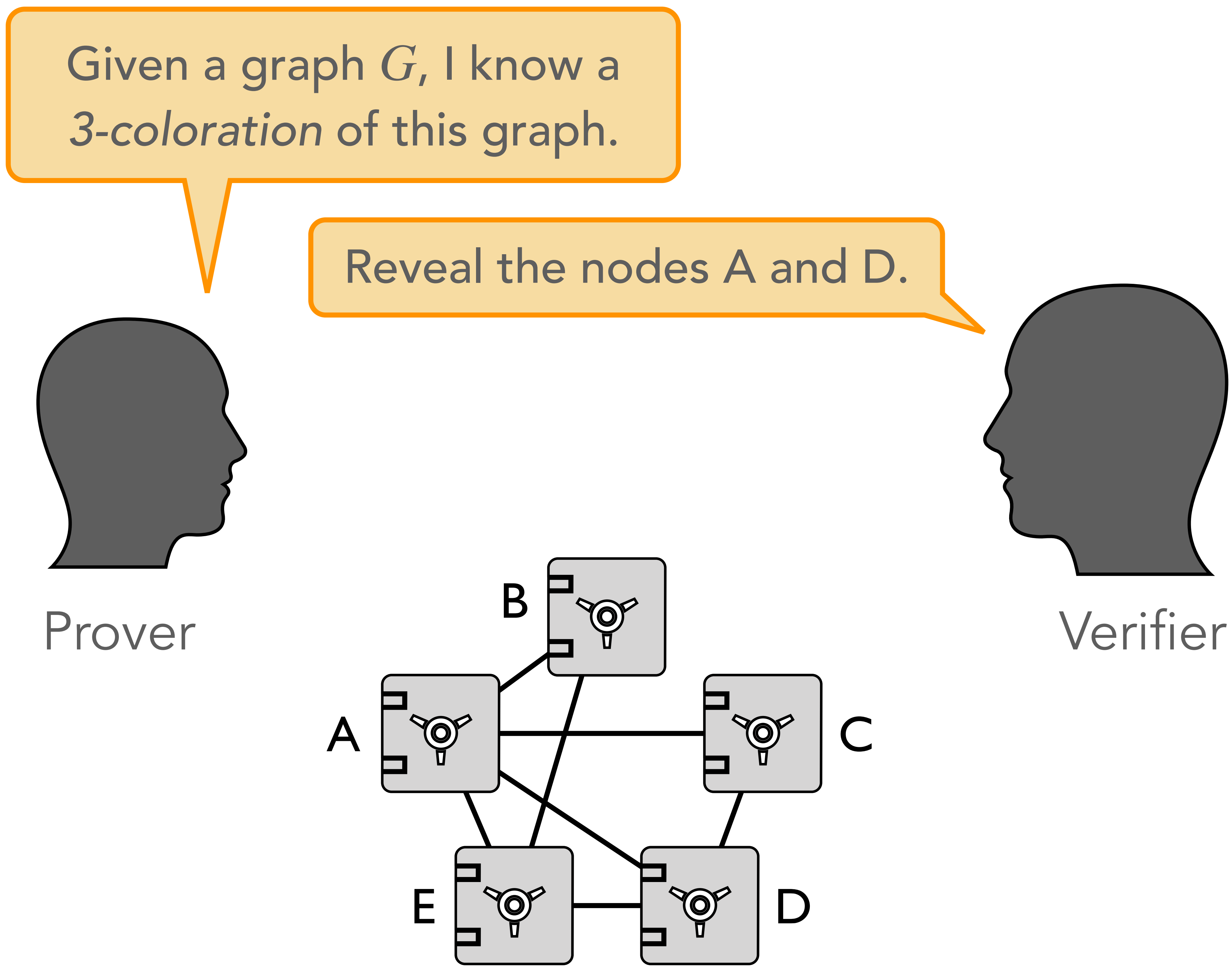


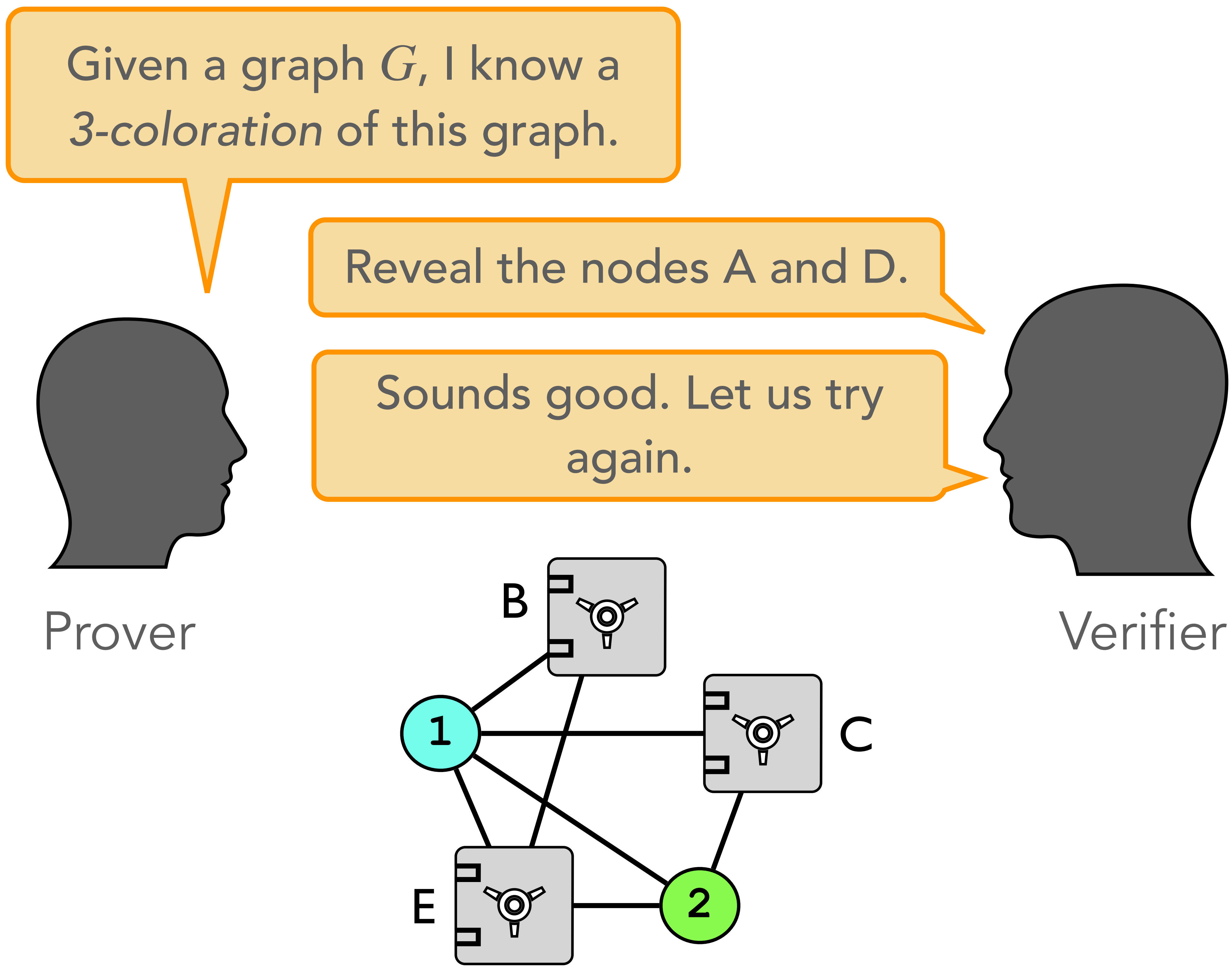
- Binding: the opened value is ensured to be committed data x.
- Hiding:  leaks no information about the committed data x (without ).

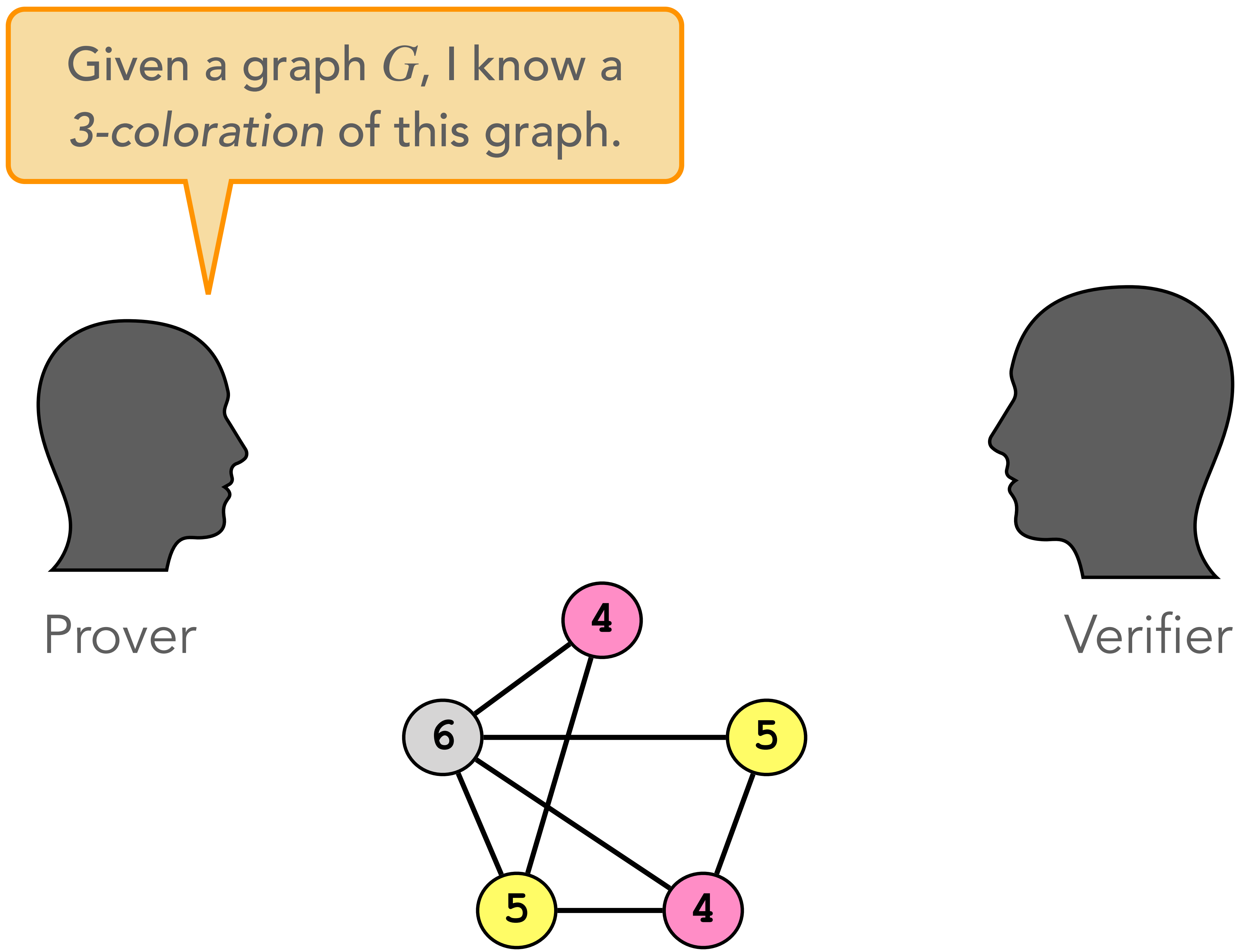

Given a graph G, I know a *3-coloration* of this graph.
Prover
Verifier

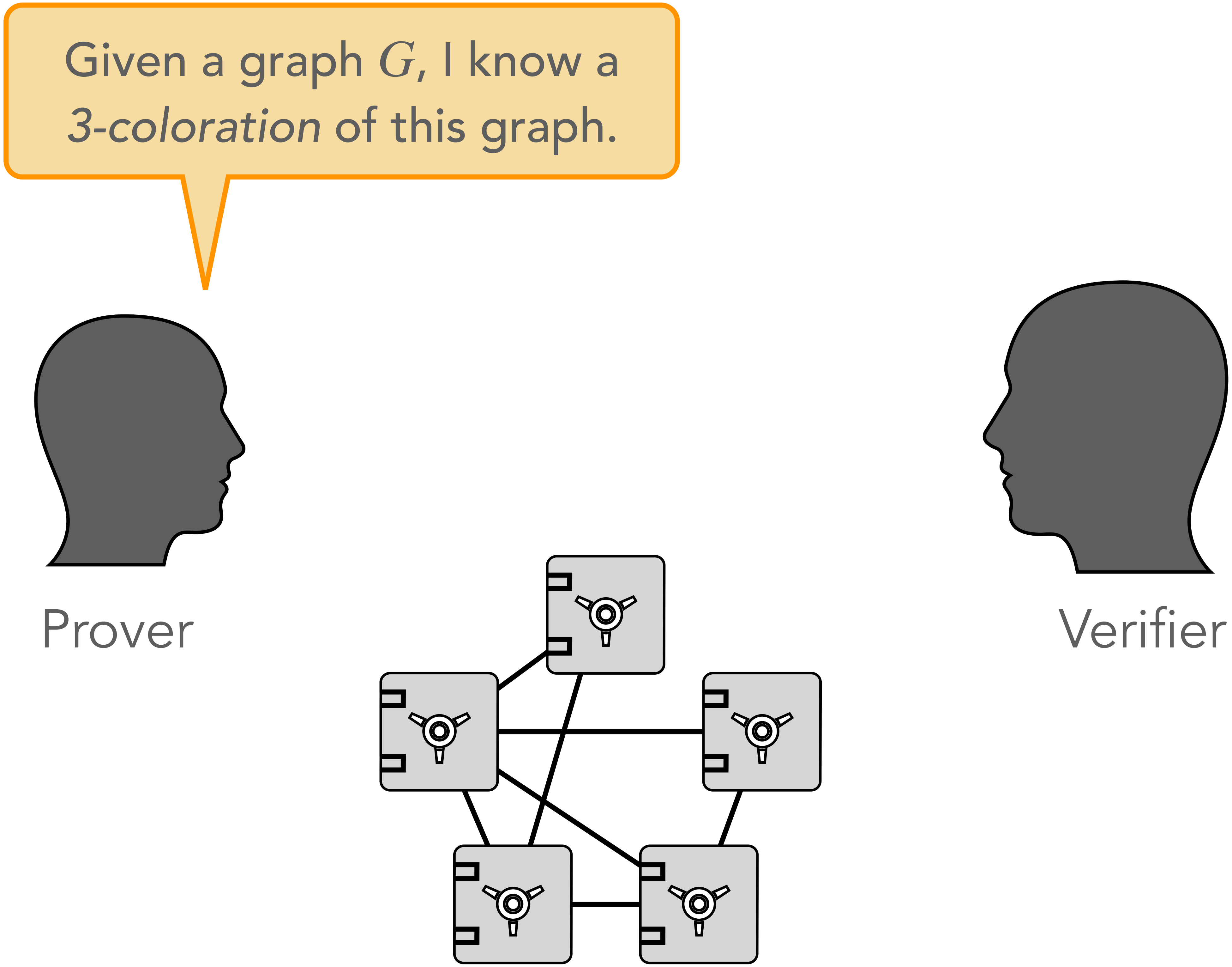

Given a graph G, I know a *3-coloration* of this graph.
Prover
Verifier
2
1
3
3
2

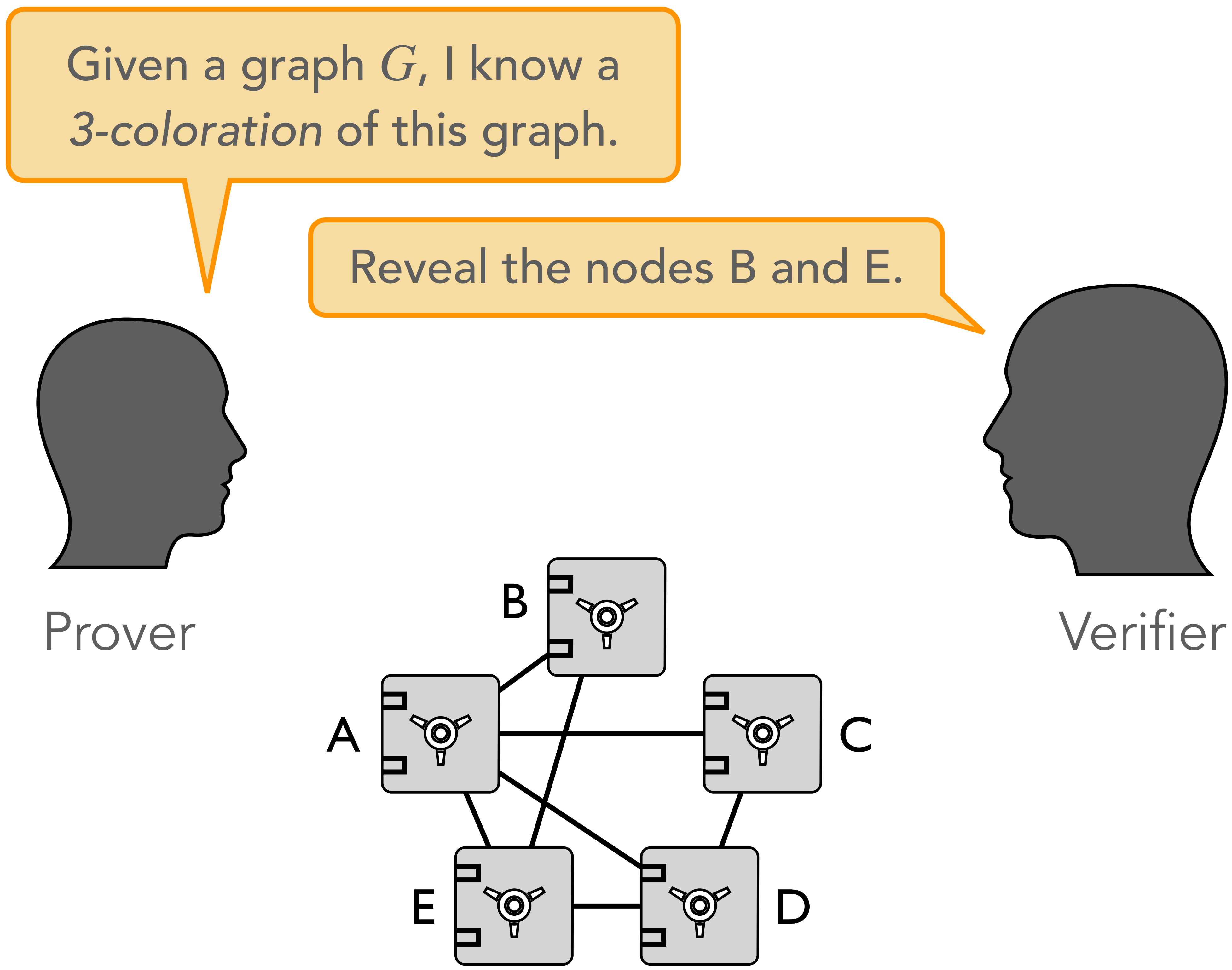

Given a graph G, I know a *3-coloration* of this graph.
Prover
Verifier
Commitment Scheme

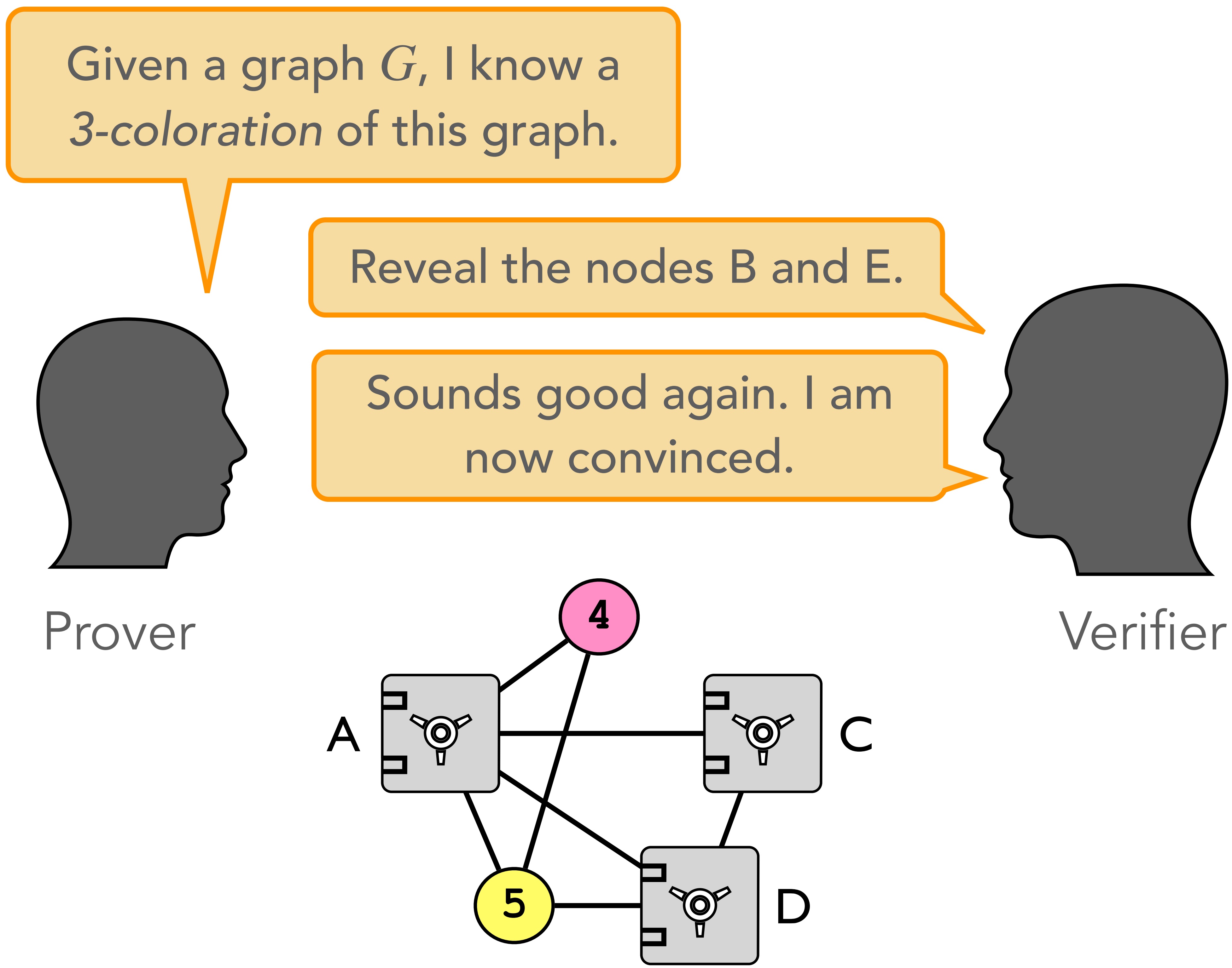

Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes A and D.
Prover
Verifier
A
B
C
D
E


Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes A and D.
Sounds good. Let us try again.
Prover
Verifier
B
C
E
1
2

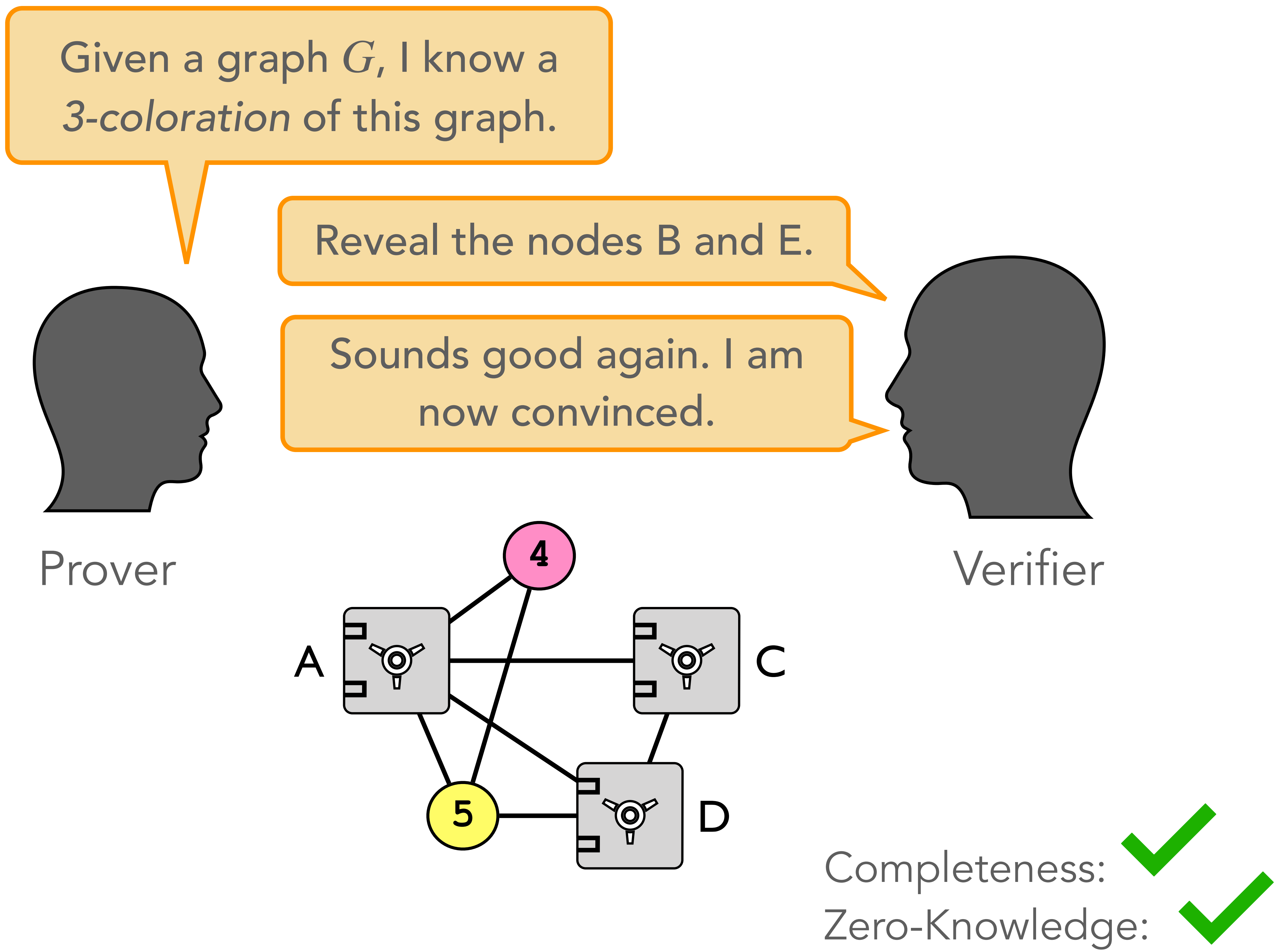

Given a graph G, I know a
3-coloration of this graph.
Prover
Verifier
4
6
5
5
4

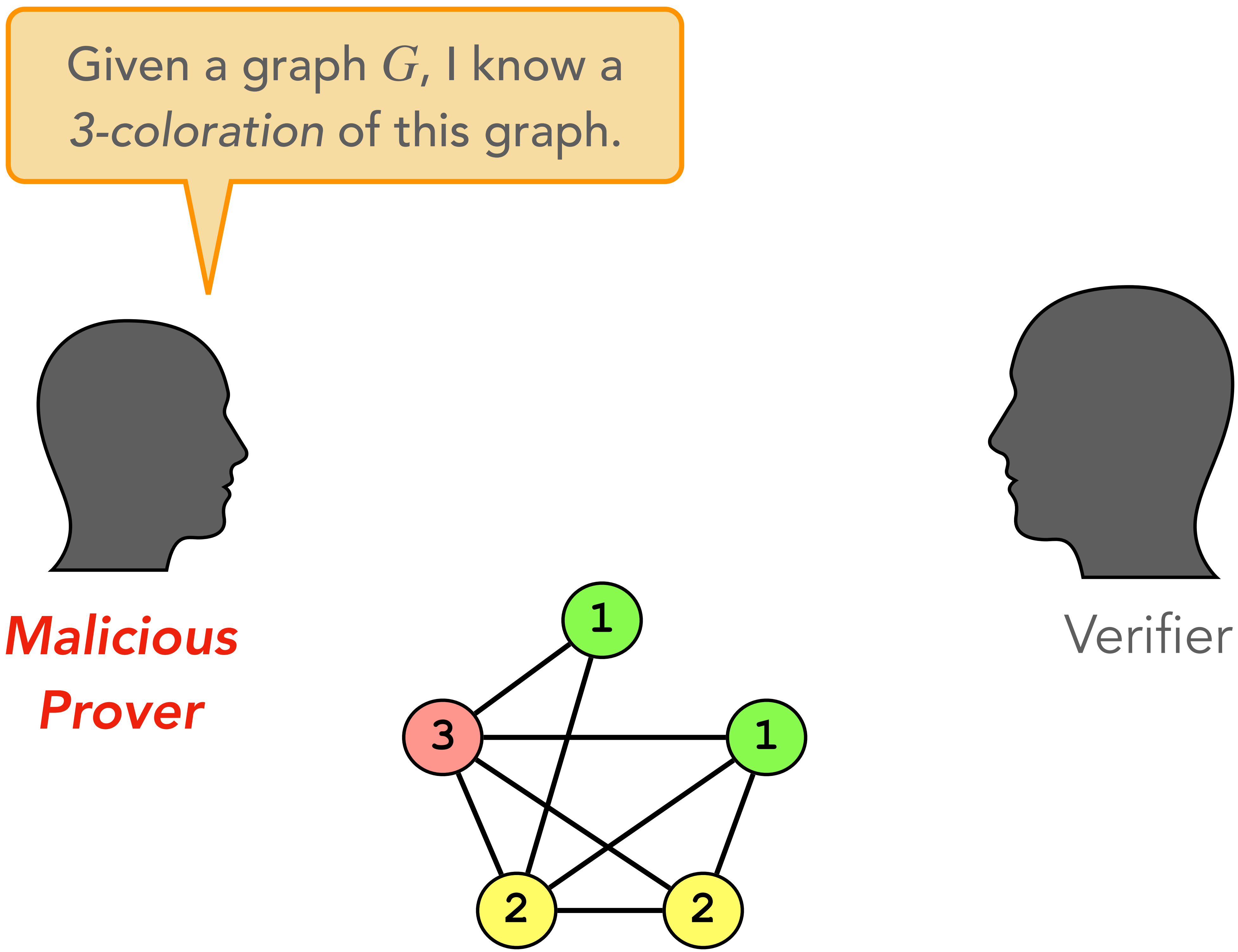

Given a graph G, I know a *3-coloration* of this graph.
Prover
Verifier

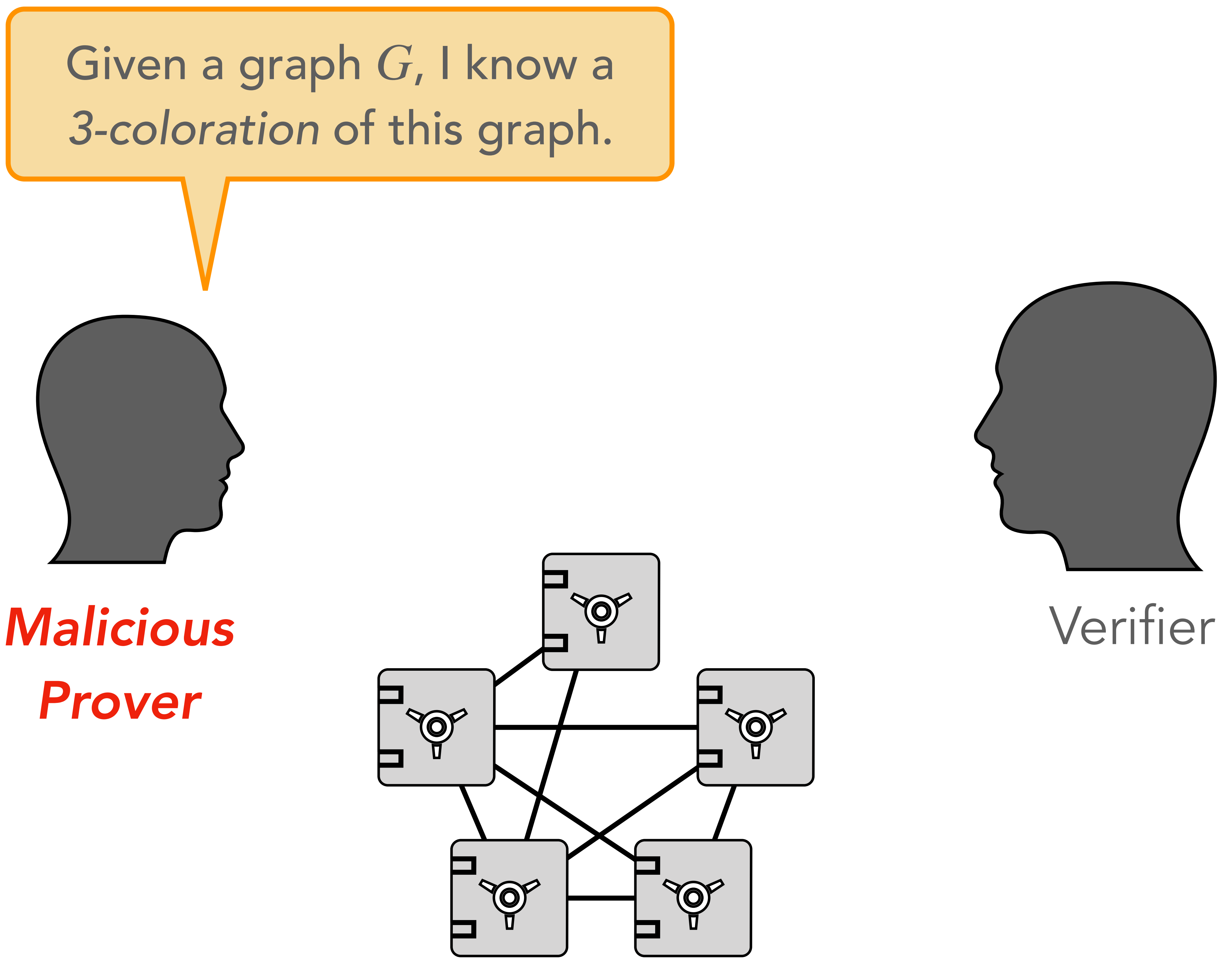

Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes B and E.
Prover
Verifier
A
B
C
D
E

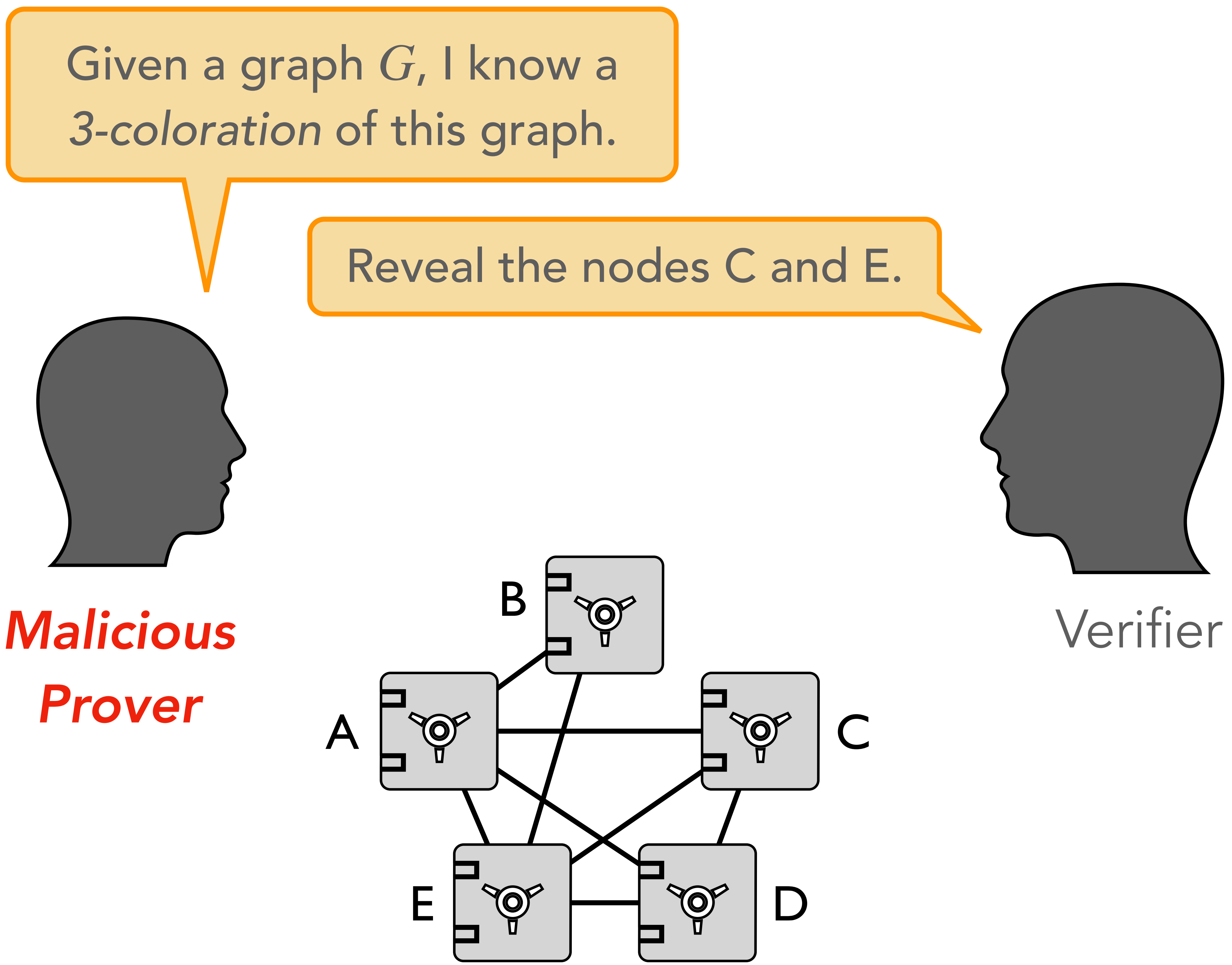


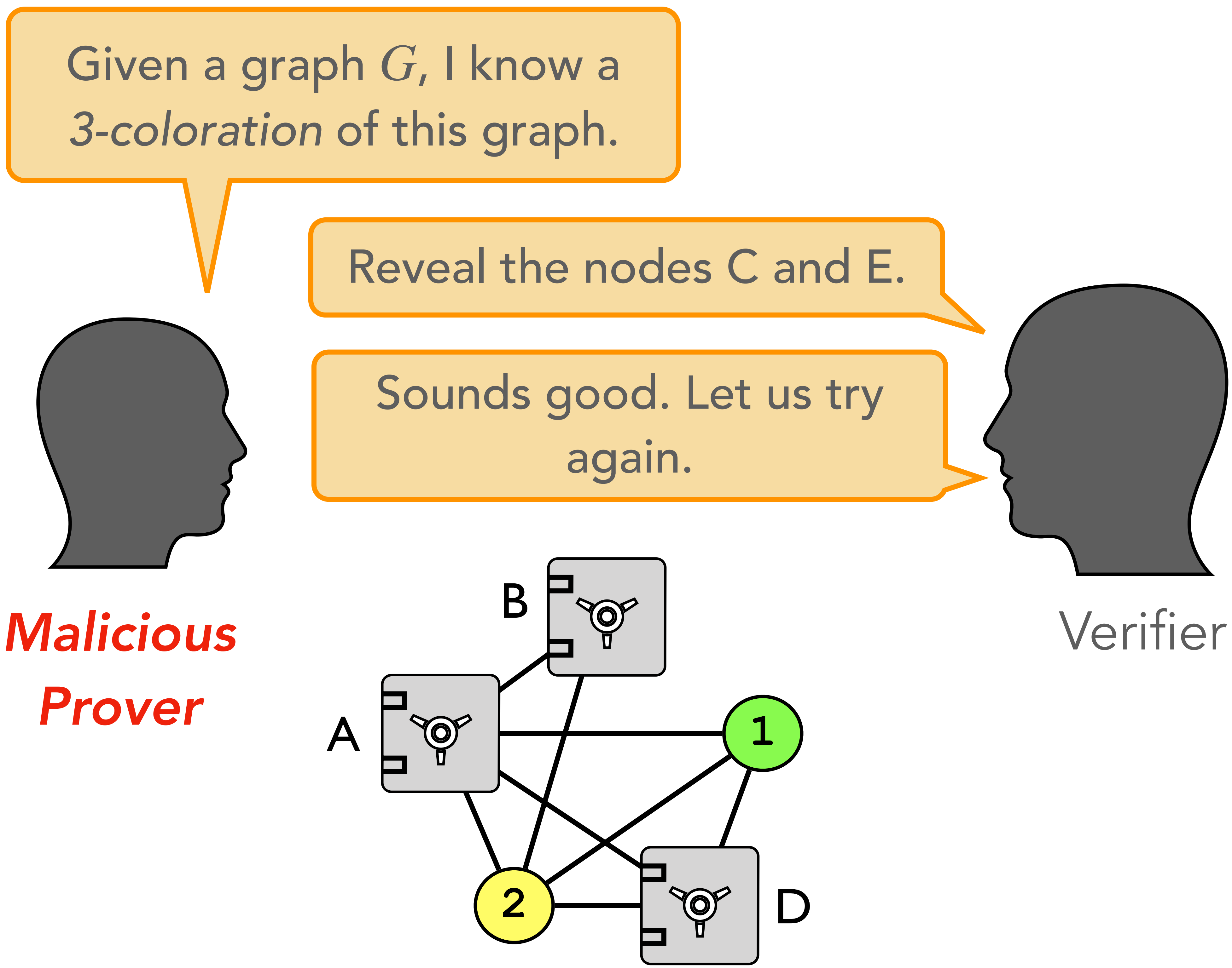
Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes B and E.
Sounds good again. I am now convinced.
Prover
Verifier
4
A
C
5
D

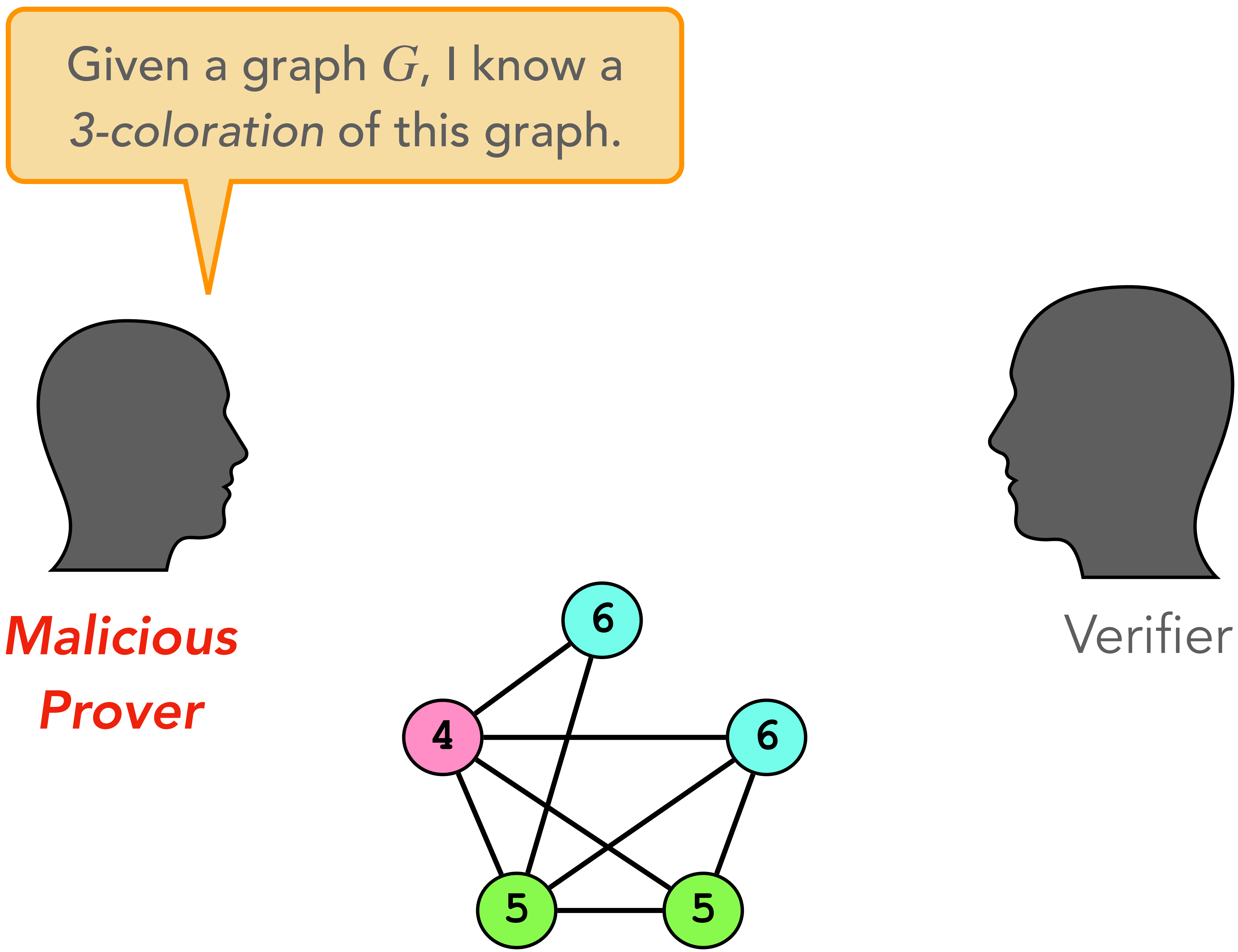

Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes B and E.
Sounds good again. I am now convinced.
Prover
Verifier
4
A
C
5
D
Completeness:
Zero-Knowledge:


Given a graph G, I know a
3-coloration of this graph.
Malicious
Prover
Verifier
1
3
1
2
2

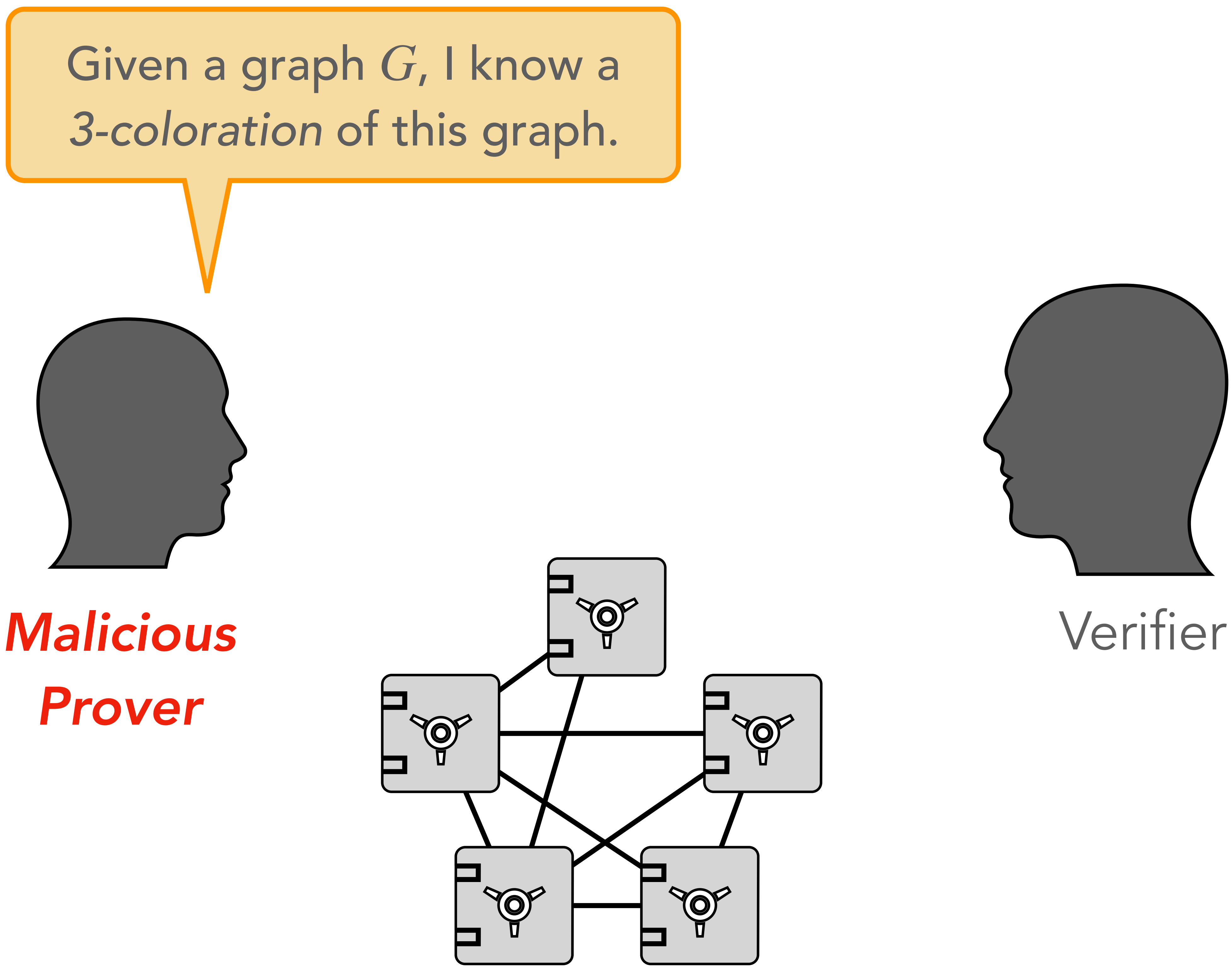

Given a graph G, I know a *3-coloration* of this graph.
Malicious Prover
Verifier

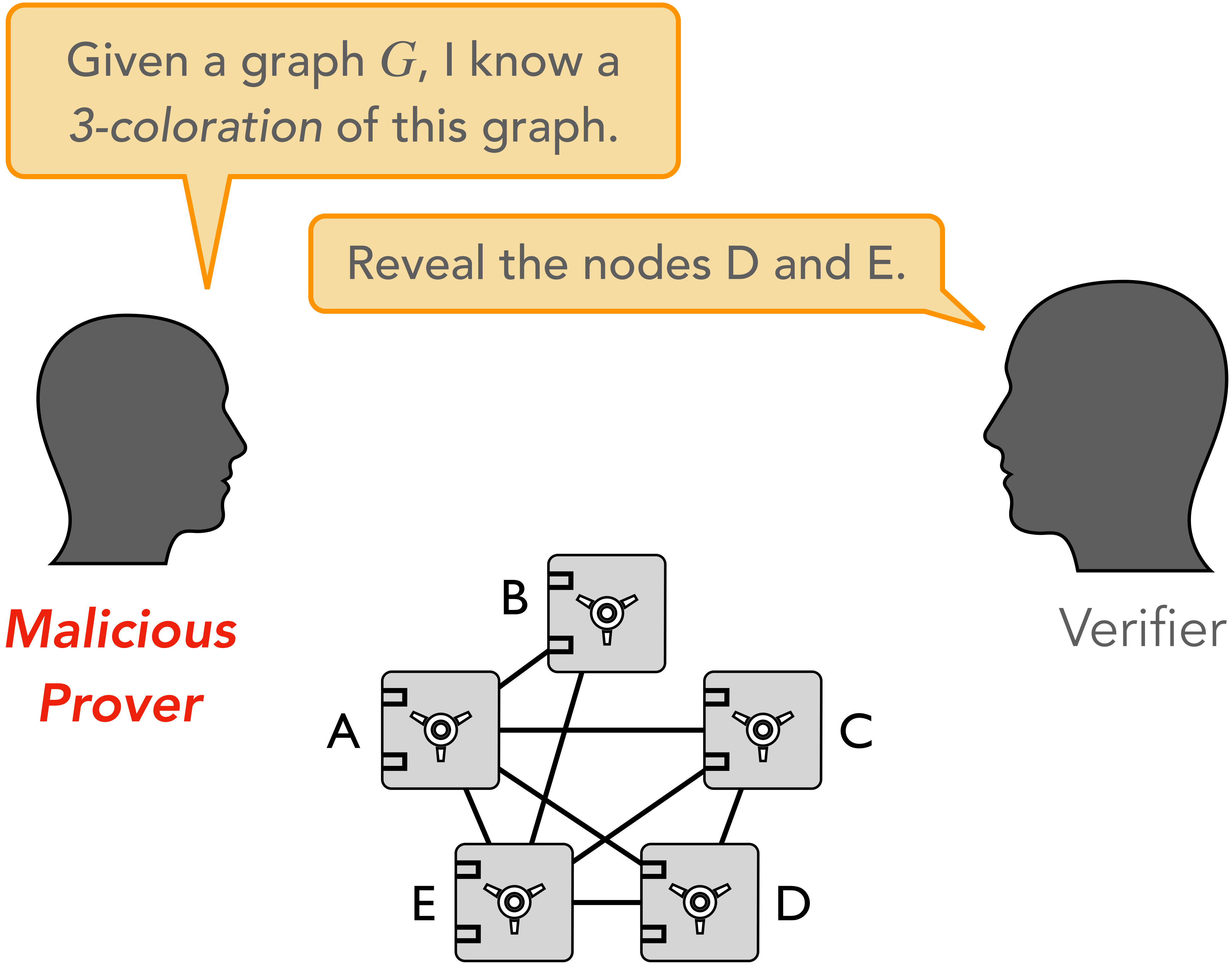

Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes C and E.
Malicious Prover
Verifier
B
A
C
E
D

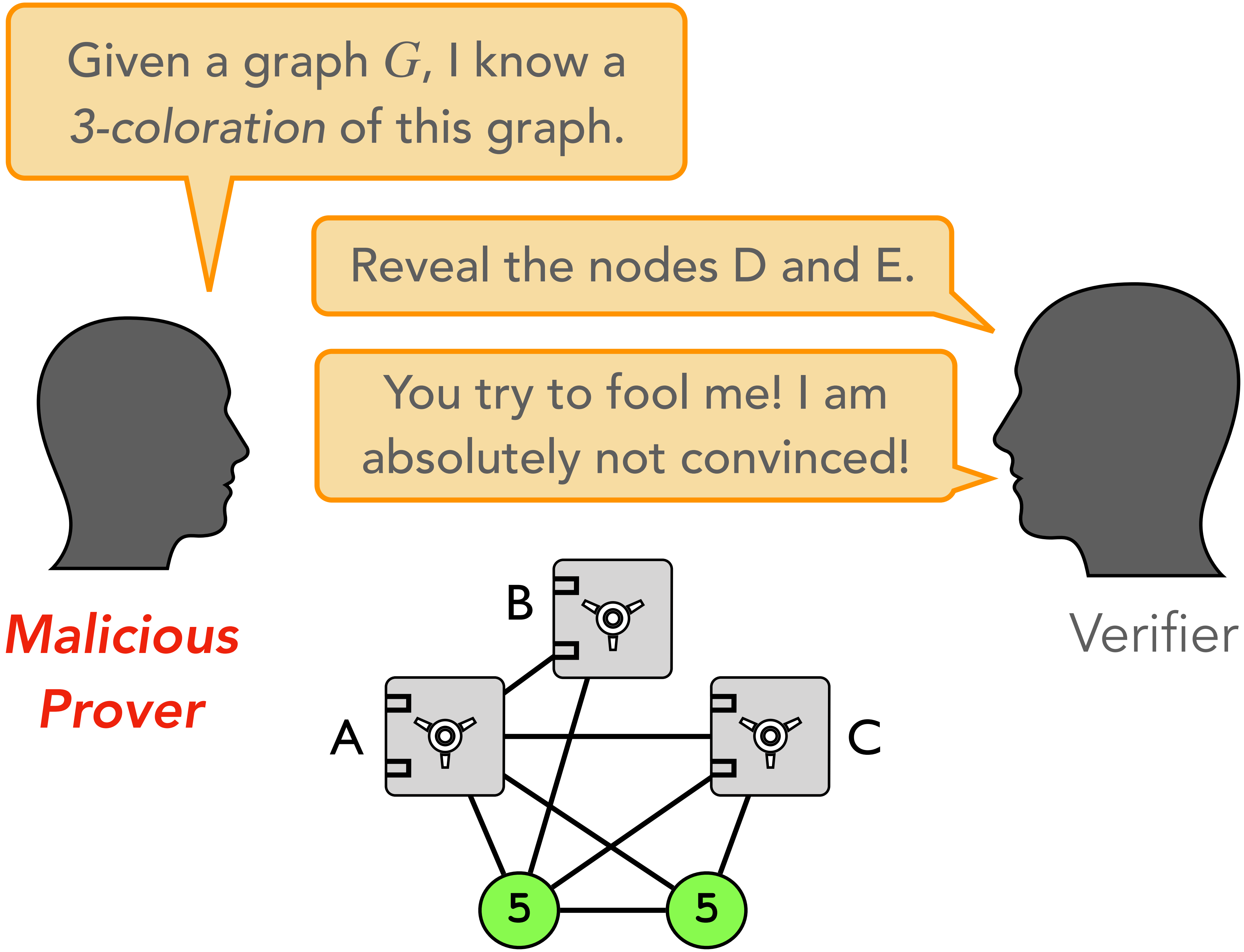

Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes C and E.
Sounds good. Let us try again.
Malicious Prover
Verifier
B
A
1
2
D



Given a graph G, I know a *3-coloration* of this graph.
Malicious Prover
Verifier
6
4
6
5
5


Given a graph G, I know a *3-coloration* of this graph.
Malicious Prover
Verifier


Given a graph G, I know a *3-coloration* of this graph.
Reveal the nodes D and E.
Malicious Prover
Verifier
B
A
C
E
D



Probability to catch a malicious prover $\geq \frac{1}{\text{nb edges}}$

Probability ε to **fail to detect** a malicious prover:

With 1 try: $\varepsilon \leq 1 - \dfrac{1}{\text{nb edges}}$

With 2 tries: $\varepsilon \leq \left(1 - \dfrac{1}{\text{nb edges}}\right)^2$

$\ldots$

With τ tries: $\varepsilon \leq \left(1 - \dfrac{1}{\text{nb edges}}\right)^\tau$

Probability ε to **fail to detect** a malicious prover:

With 1 try: $\varepsilon \leq 1 - \dfrac{1}{\text{nb edges}}$

With 2 tries: $\varepsilon \leq \left(1 - \dfrac{1}{\text{nb edges}}\right)^2$

$\ldots$

With τ tries: $\varepsilon \leq \left(1 - \dfrac{1}{\text{nb edges}}\right)^\tau$

For a graph with 100 edges

1 try: 99.00 %

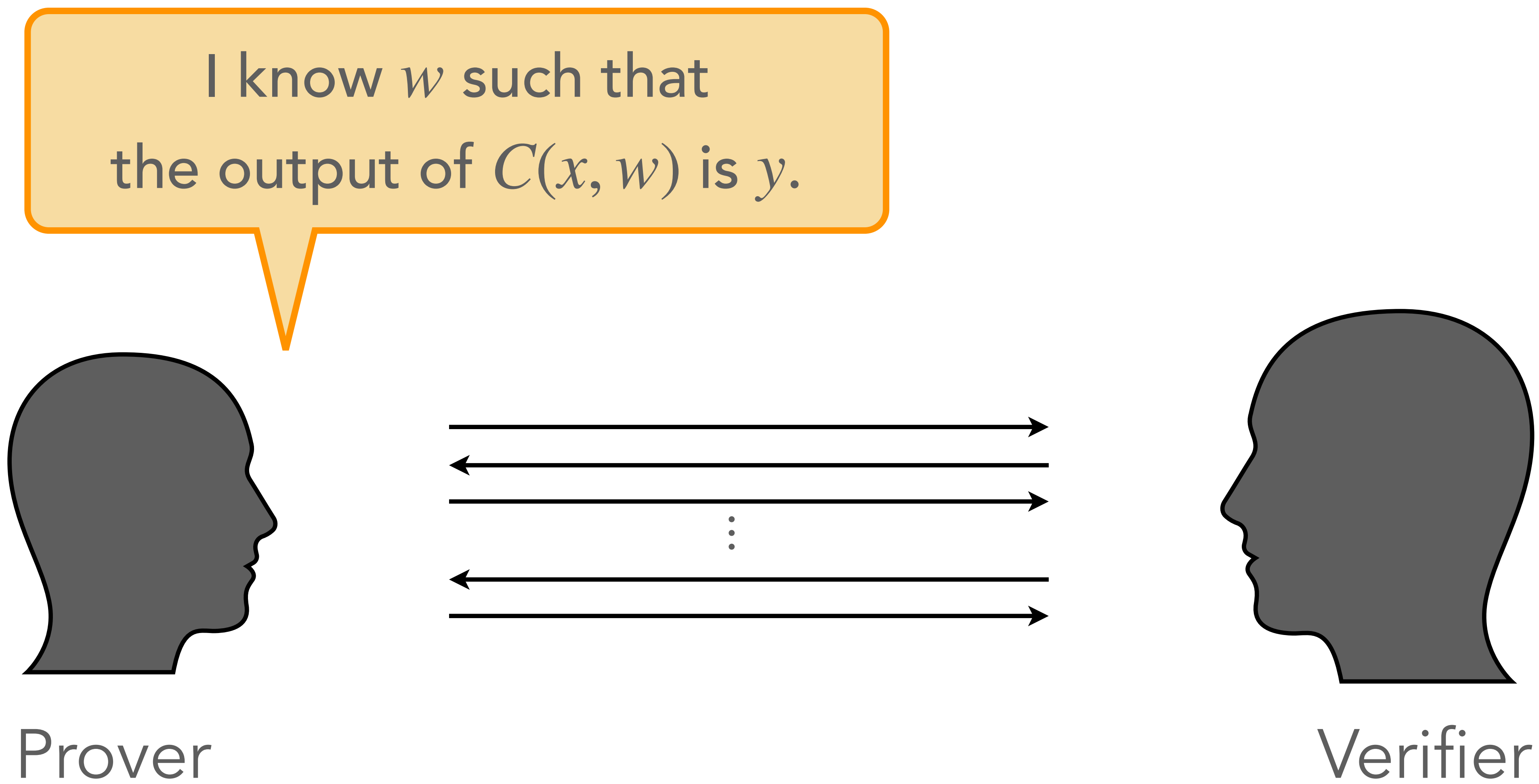
2 tries: 98.01 %

3 tries: 97.03 %

100 tries: 36.60 %

458 tries: 1.00 %

1000 tries: 0.004 %

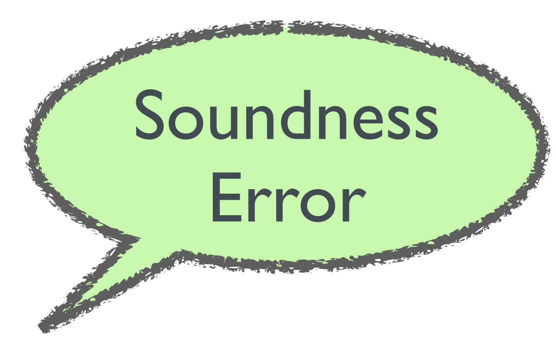


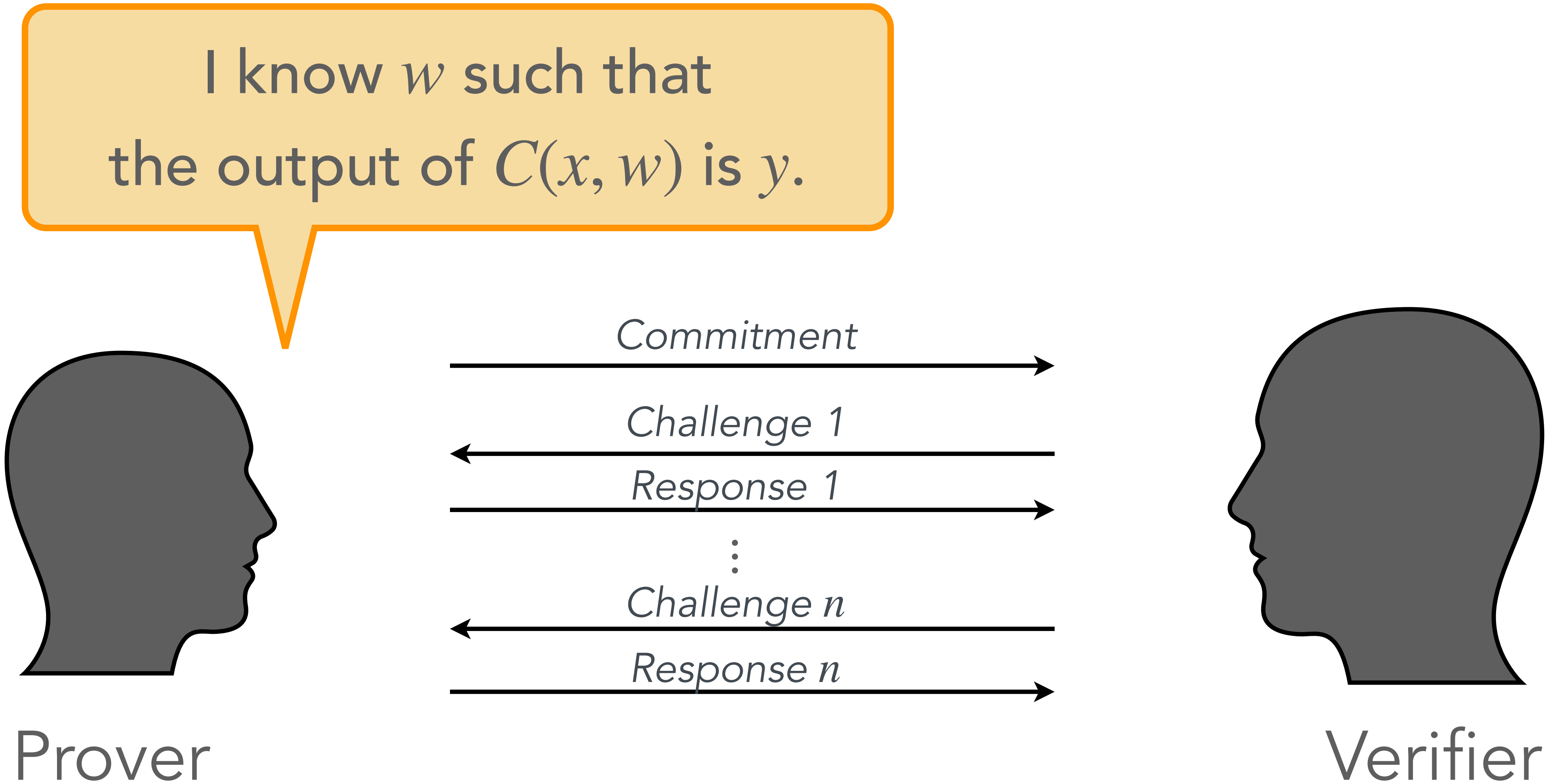
- **<u>Perfect</u>** <u>Completeness</u>:

$$\mathrm{Prob}\left[\text{verifier convinced} \,|\, \text{prover honest}\right] = 1$$

- **<u>Statistical</u>** <u>Soundness</u>:

$$\mathrm{Prob}\left[\text{verifier convinced} \,|\, \text{prover malicious}\right] \leq \varepsilon$$



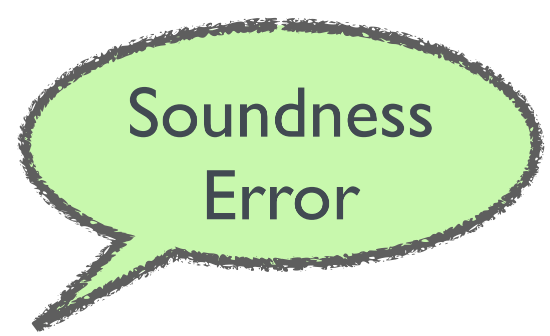


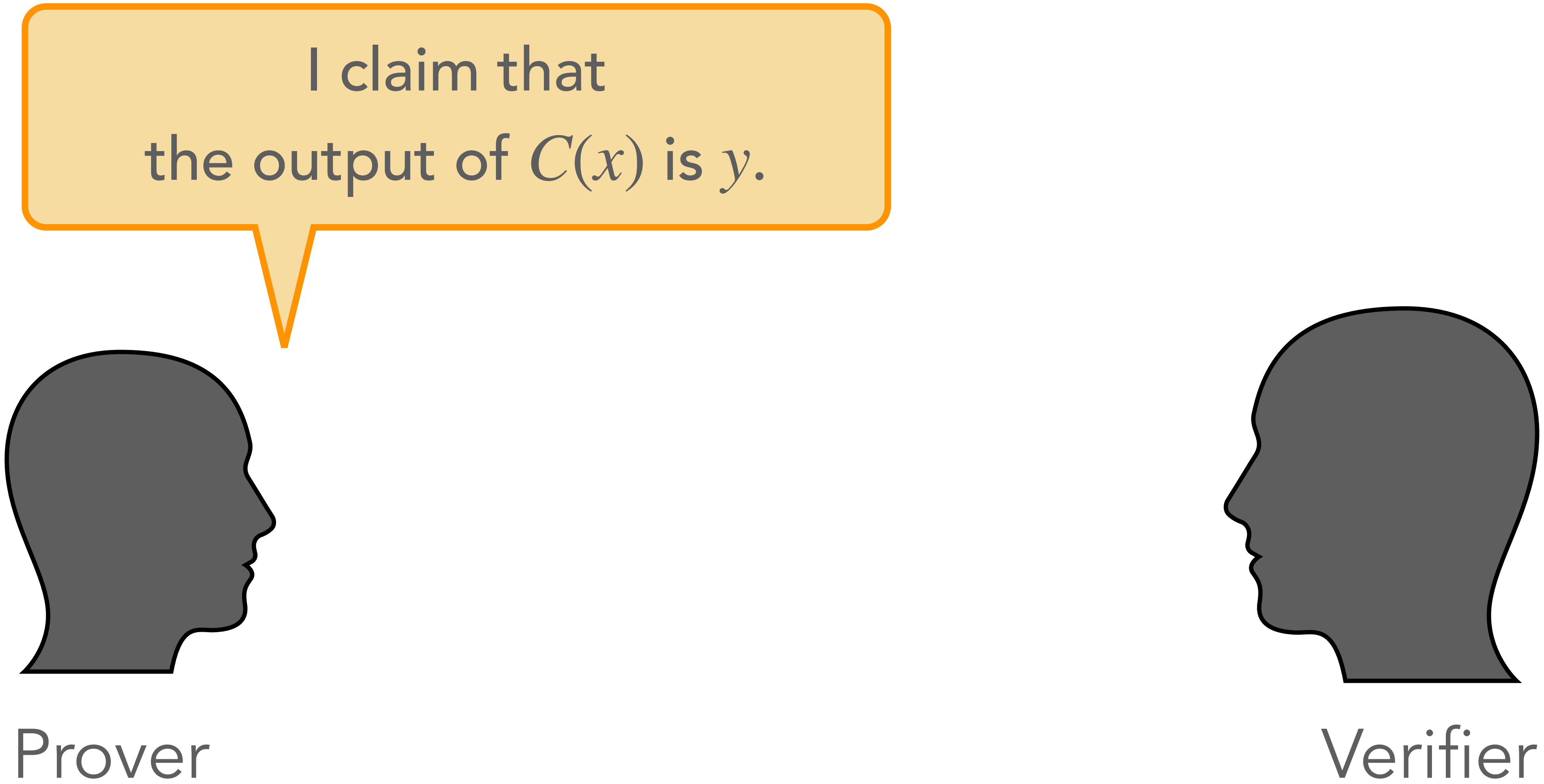
- **Perfect** Completeness:

$$\text{Prob}\left[\text{verifier convinced} \,|\, \text{prover honest}\right] = 1$$

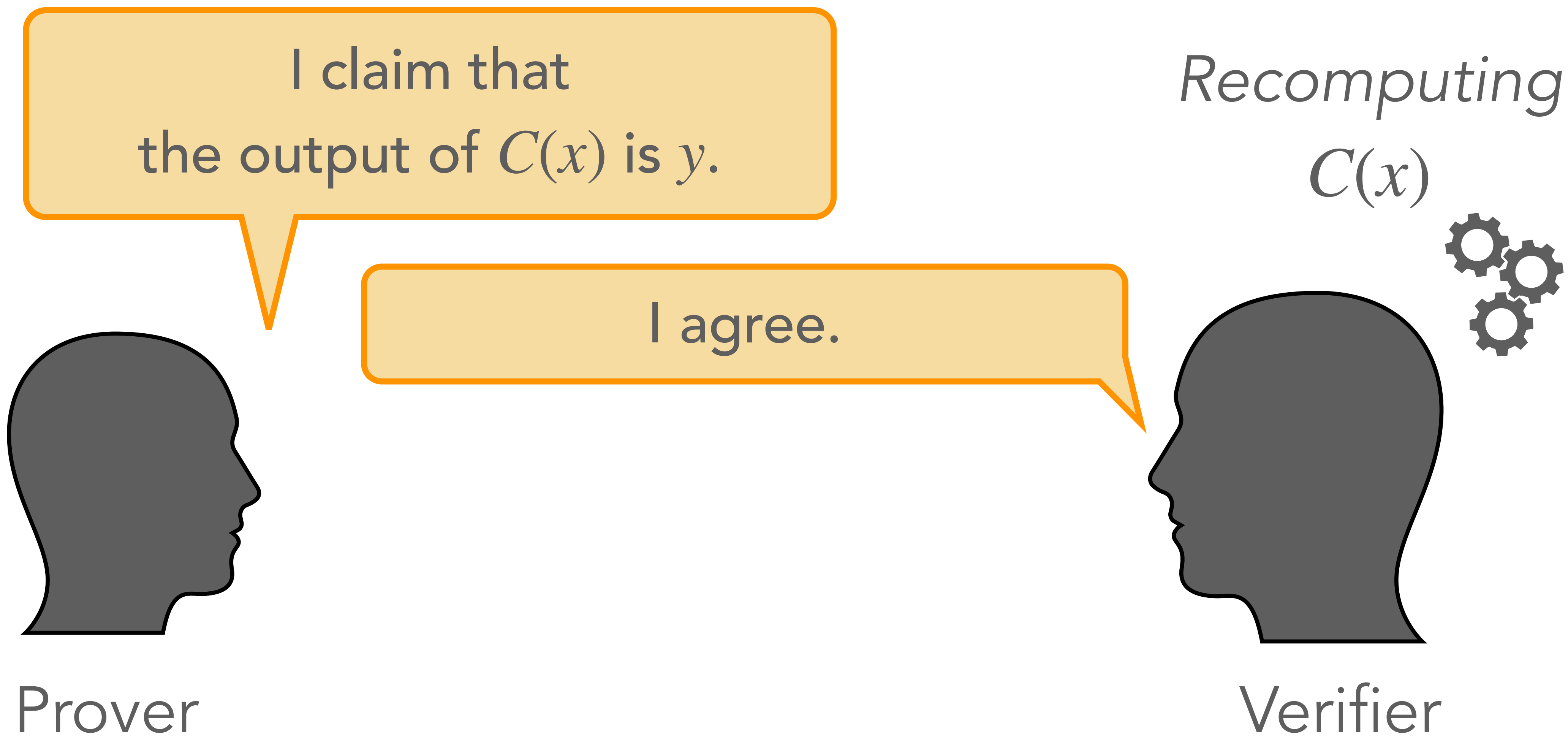
- **Statistical** Soundness:

$$\text{Prob}\left[\text{verifier convinced} \,|\, \text{prover malicious}\right] \leq \varepsilon$$

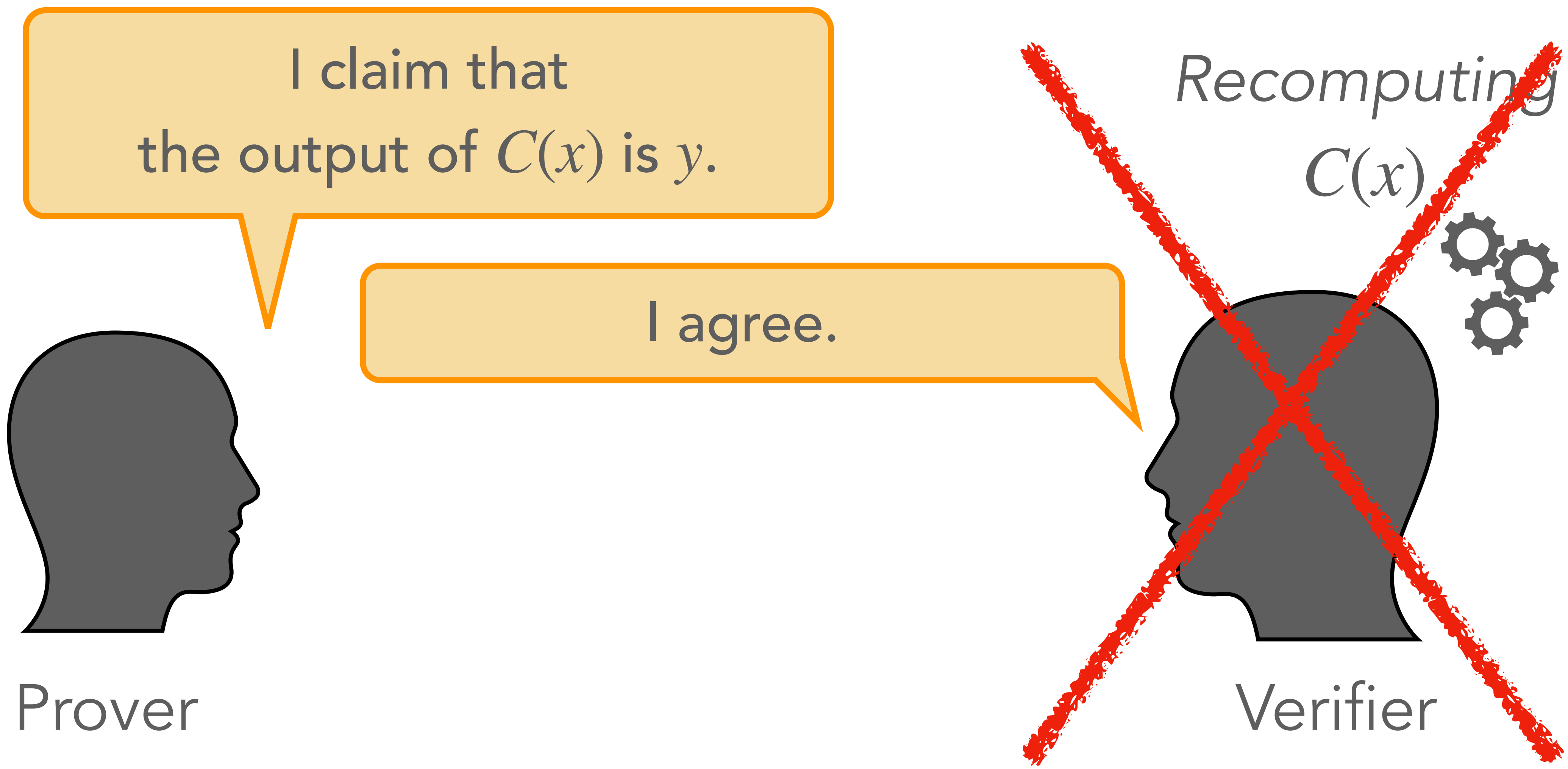




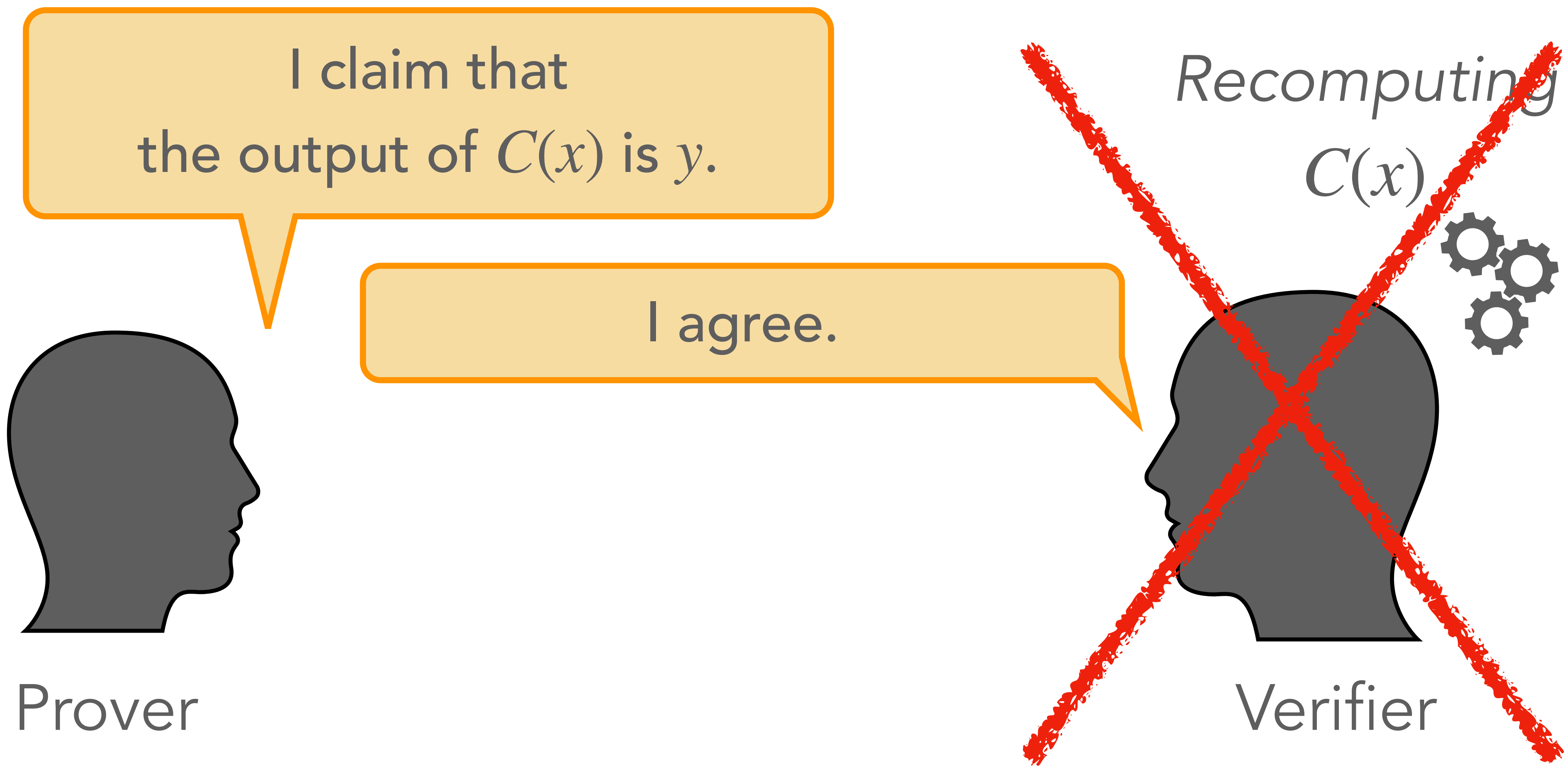
- I claim that the number of occurrences of the word “mais” in the book “A la Recherche du Temps Perdu” written by Marcel Proust is 8256.
  - x is the text of the book
  - C is the counting algorithm for the word “mais”.
  - y is the number 8256



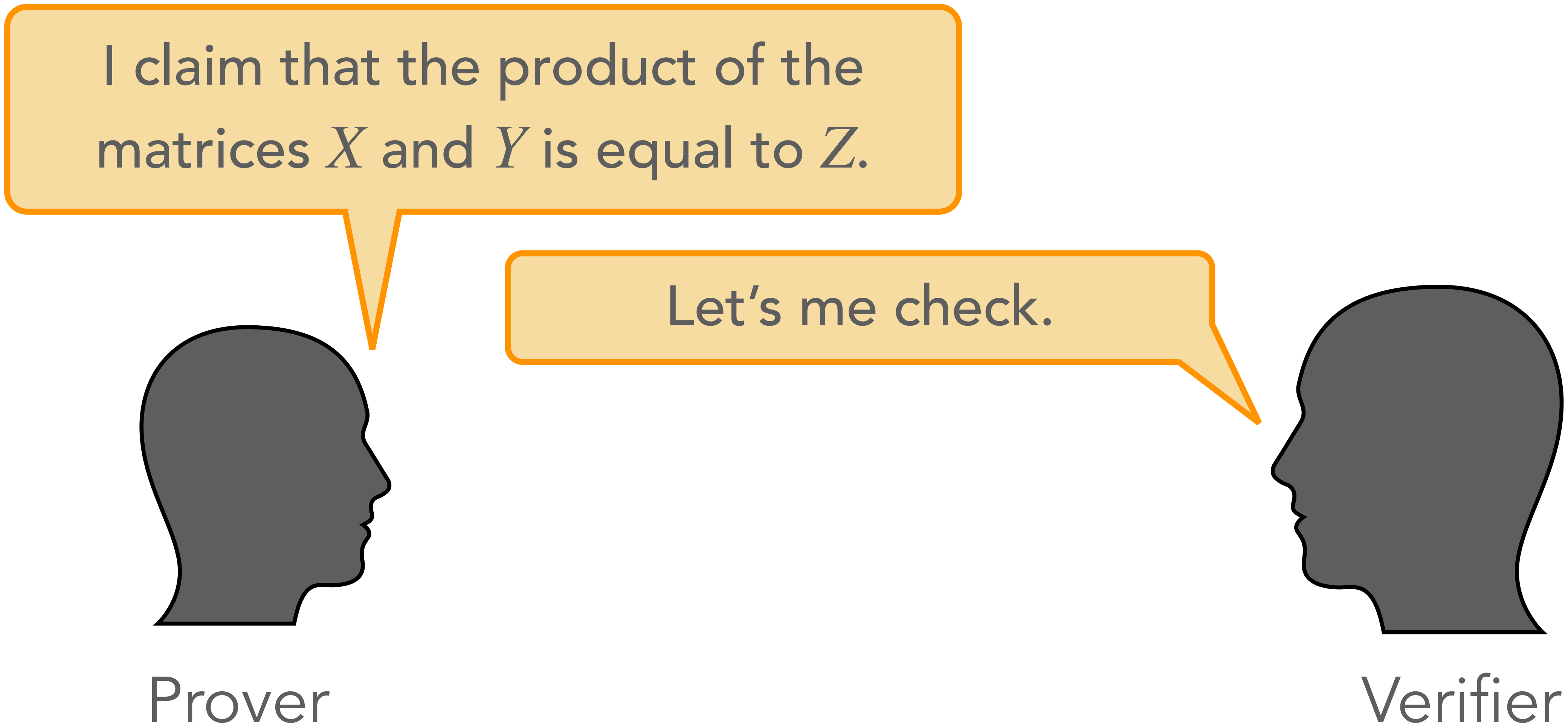
- I claim that the number of occurrences of the word "mais" in the book "A la Recherche du Temps Perdu" written by Marcel Proust is 8256.
  - x is the text of the book
  - C is the counting algorithm for the word "mais".
  - y is the number 8256



- I claim that the number of occurrences of the word "mais" in the book "A la Recherche du Temps Perdu" written by Marcel Proust is 8256.
  - x is the text of the book
  - C is the counting algorithm for the word "mais".
  - y is the number 8256



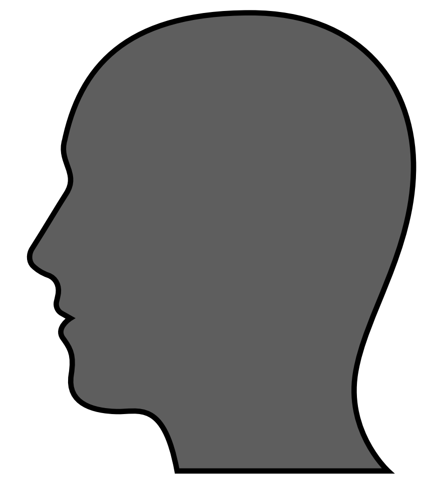
Efficiency: the verifier should ***be faster*** than the time required to compute $C(x)$.


I claim that the product of the matrices X and Y is equal to Z.
Let's me check.
Prover
Verifier

Matrices $X, Y, Z \in \mathbb{F}_q^{n \times n}$

- <u>Freivalds' Algorithm</u>
  - The verifier samples a vector $r \in \mathbb{F}_q^n$
  - The verifier computes
    $$v \leftarrow X(Yr) - Zr.$$
  - Accept the claim iff $v = 0$

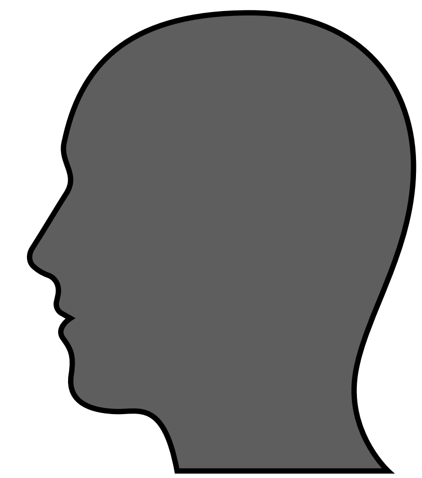
- <u>Completeness</u>
  $$v = X(Yr) - Zr = \underbrace{(XY - Z)}_{=0}\, r = 0$$



Matrices $X, Y, Z \in \mathbb{F}_q^{n\times n}$

- <u>Freivalds' Algorithm</u>
  - The verifier samples a vector $r \in \mathbb{F}_q^n$
  - The verifier computes
    $$v \leftarrow X(Yr) - Zr.$$
  - Accept the claim iff $v = 0$

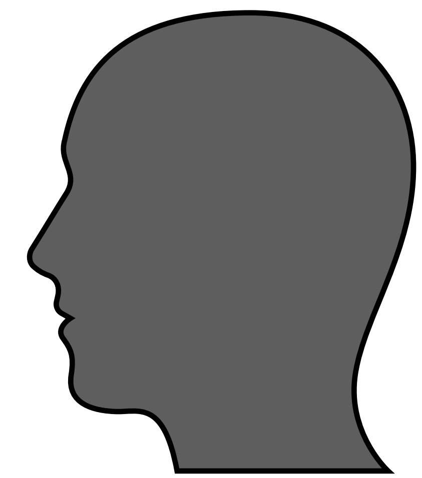
- <u>Soundness</u>
  $$v = X(Yr) - Zr = \underbrace{(XY - Z)}_{\neq 0}\, r$$



Verifier

Matrices $X, Y, Z \in \mathbb{F}_q^{n \times n}$

- Freivalds' Algorithm
  - The verifier samples a vector $r \in \mathbb{F}_q^n$
  - The verifier computes
    $$v \leftarrow X(Yr) - Zr.$$
  - Accept the claim iff $v = 0$



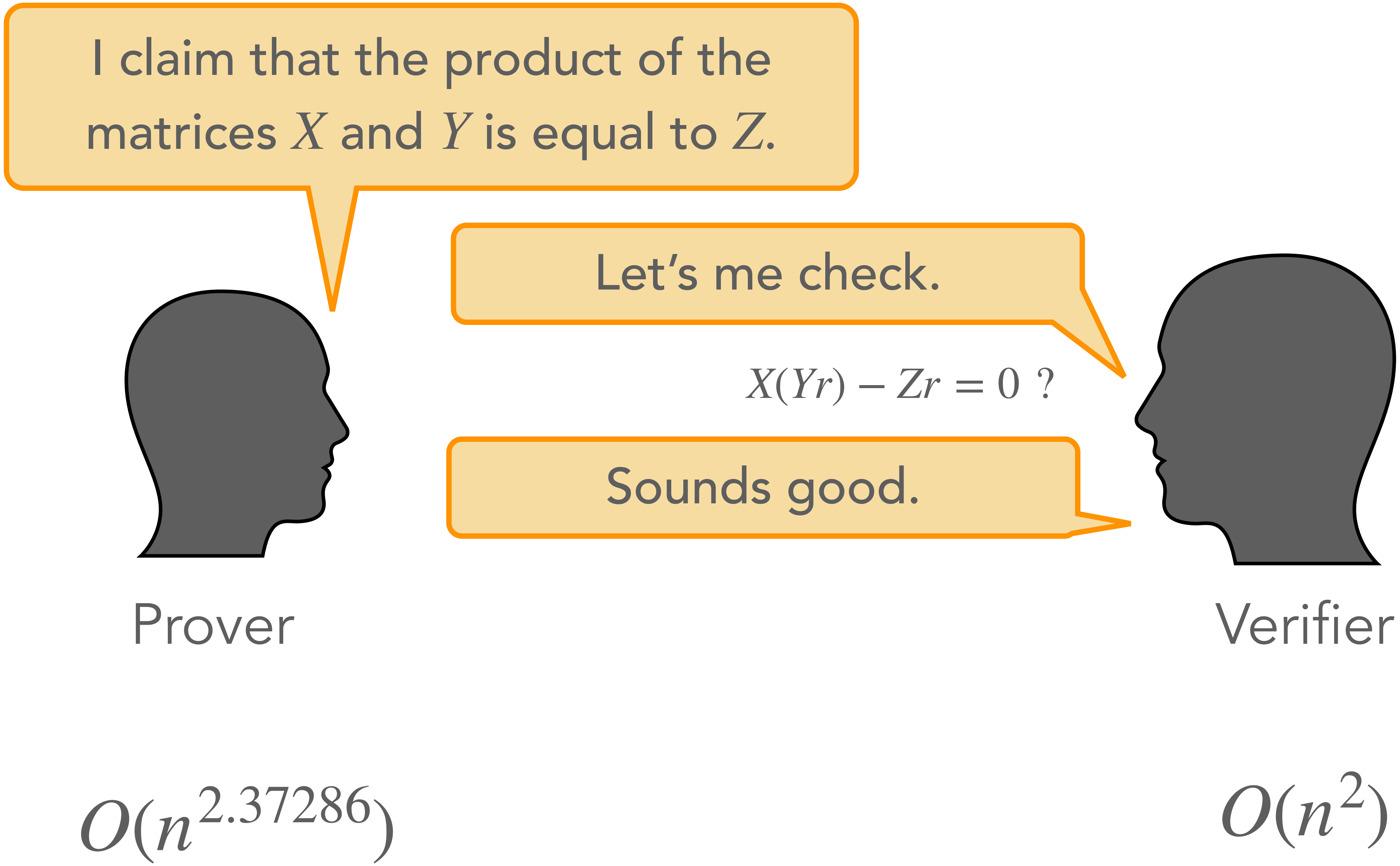
- Soundness

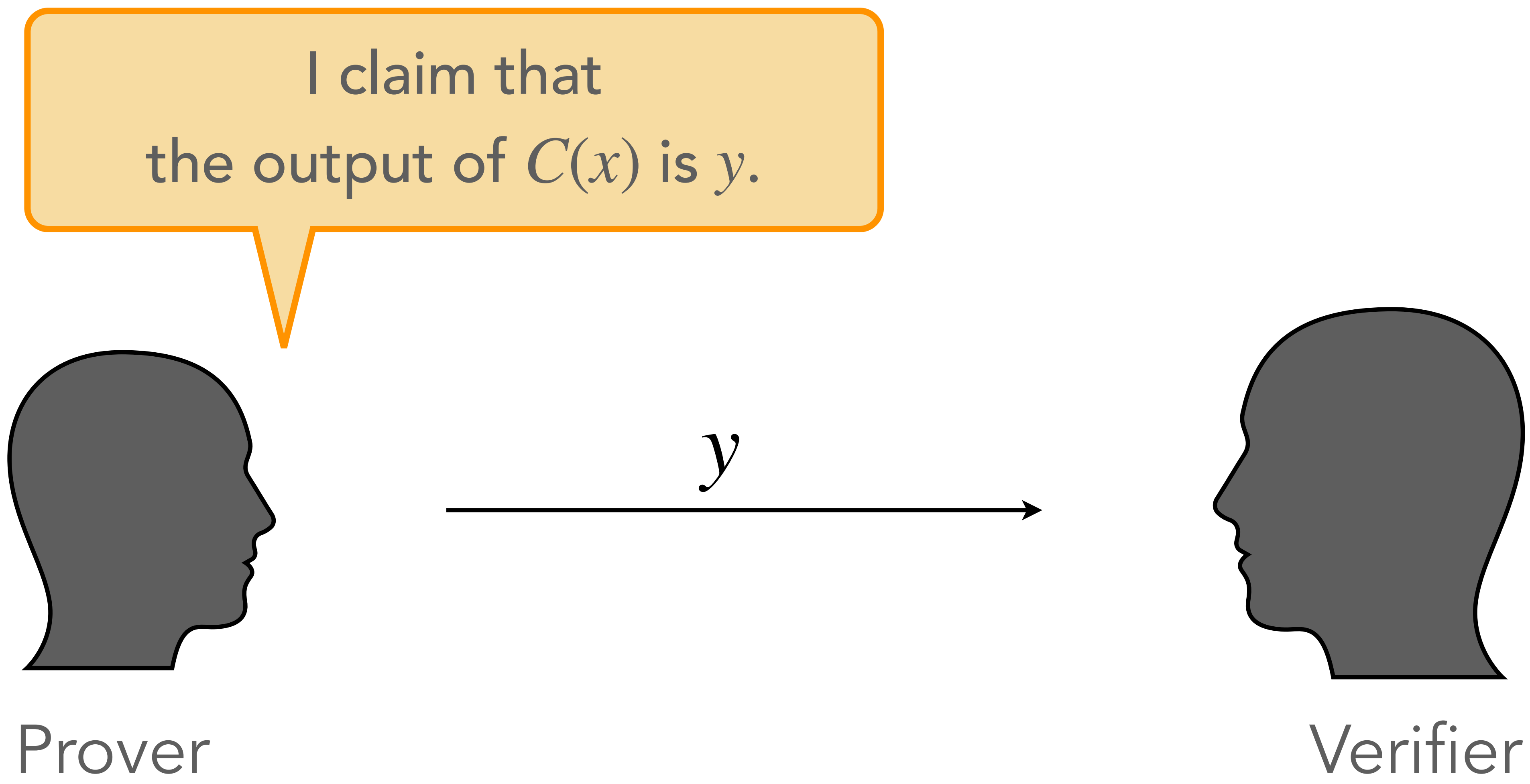
$$v = \begin{pmatrix} & \vdots & \\ \cdots & m_{i,j} & \cdots \\ & \vdots & \end{pmatrix} \begin{pmatrix} \vdots \\ r_j \\ \vdots \end{pmatrix} \text{ with } m_{i,j} \neq 0$$

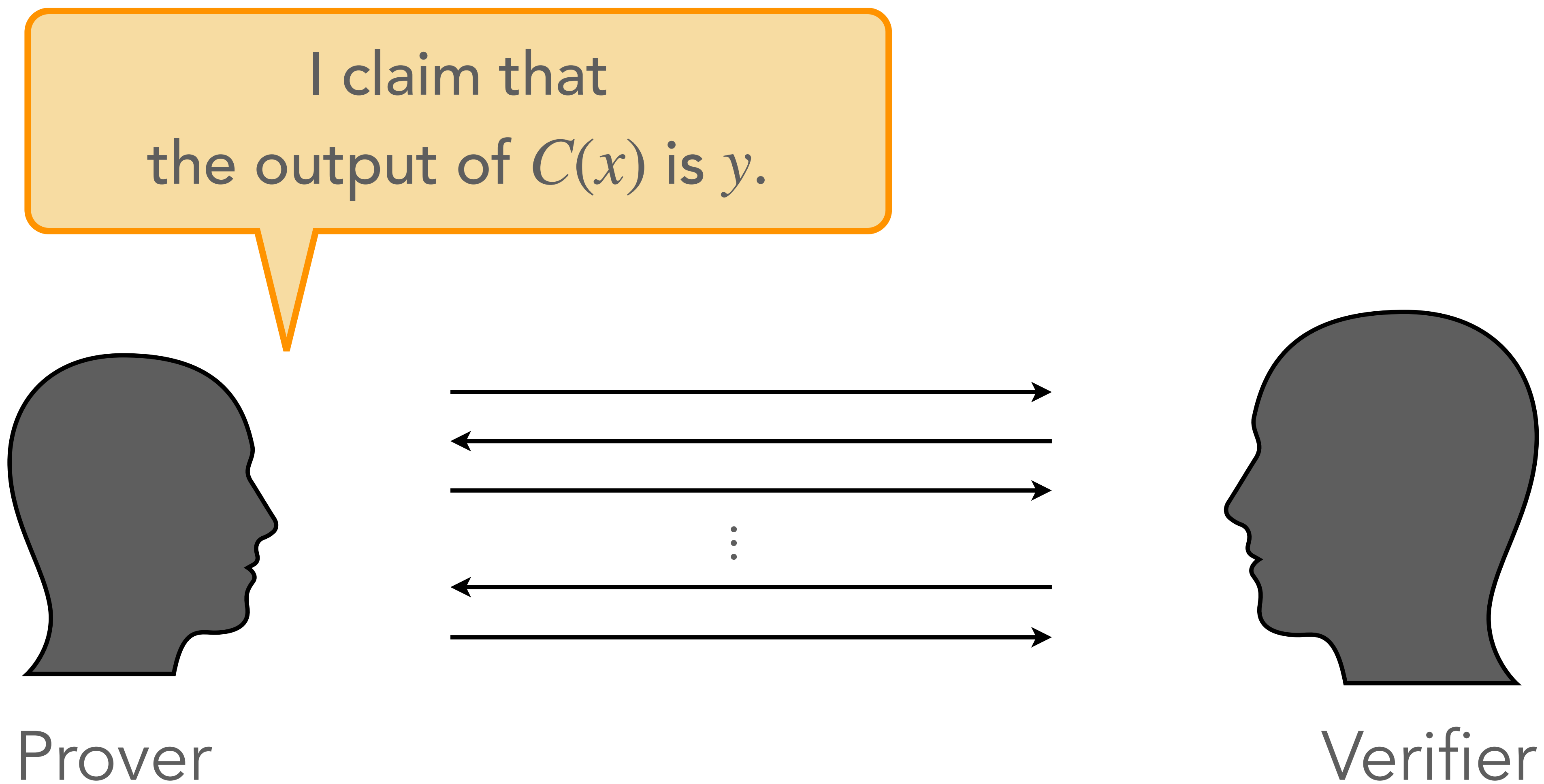
So,

$$v_i = r_j \cdot m_{i,j} + \ldots \sim \mathcal{U}(\mathbb{F}_q)$$

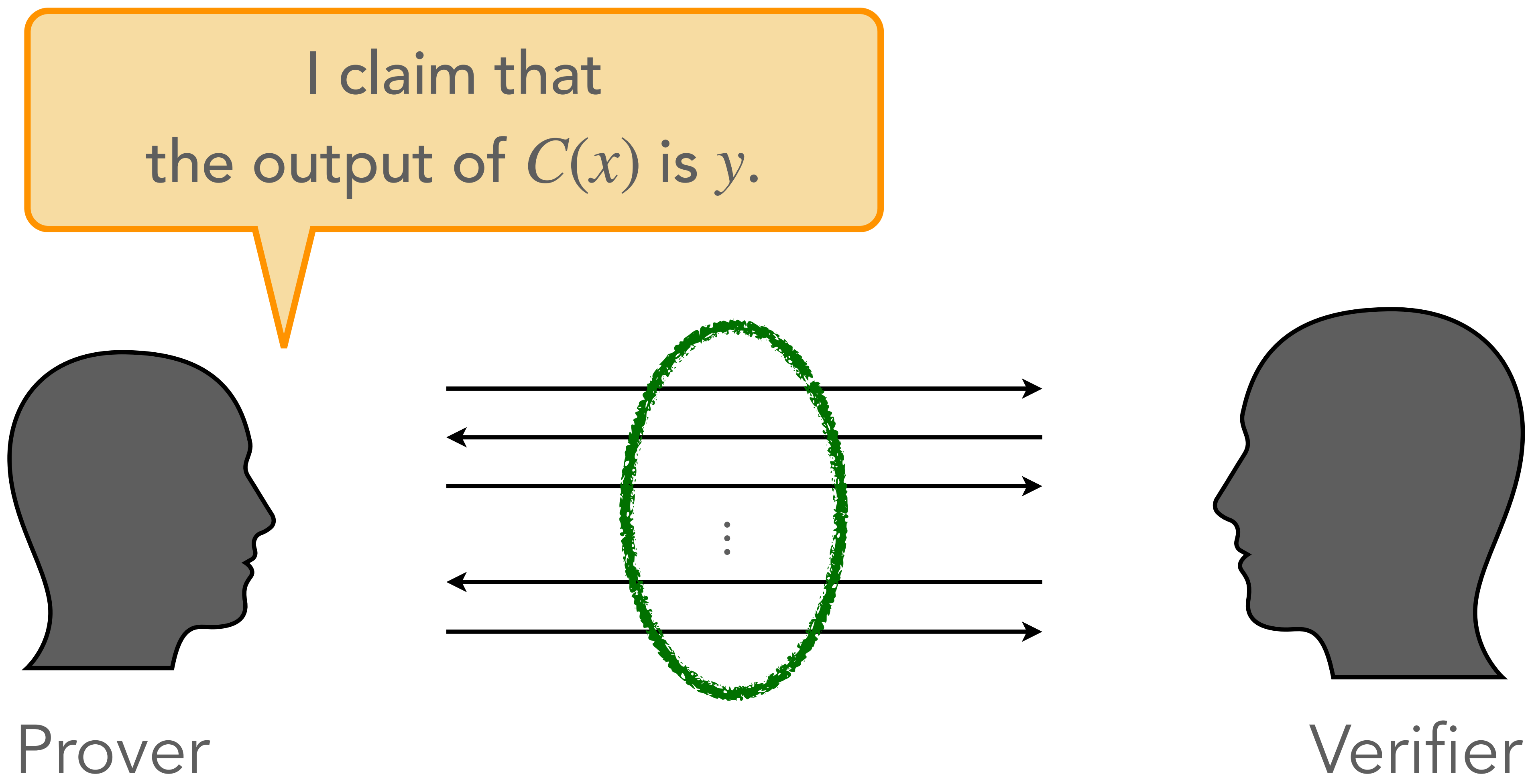
$$\text{Prob}\left[v = 0 \mid XY \neq Z\right] \geq \text{Prob}\left[v_i = 0 \mid XY \neq Z\right] = \frac{1}{q}$$


I claim that the product of the matrices X and Y is equal to Z.
Let's me check.
$X(Yr) - Zr = 0$?
Sounds good.
Prover
Verifier
$O(n^{2.37286})$
$O(n^2)$


I claim that
the output of $C(x)$ is y.
y
Prover
Verifier



I claim that
the output of $C(x)$ is y.
Prover
Verifier



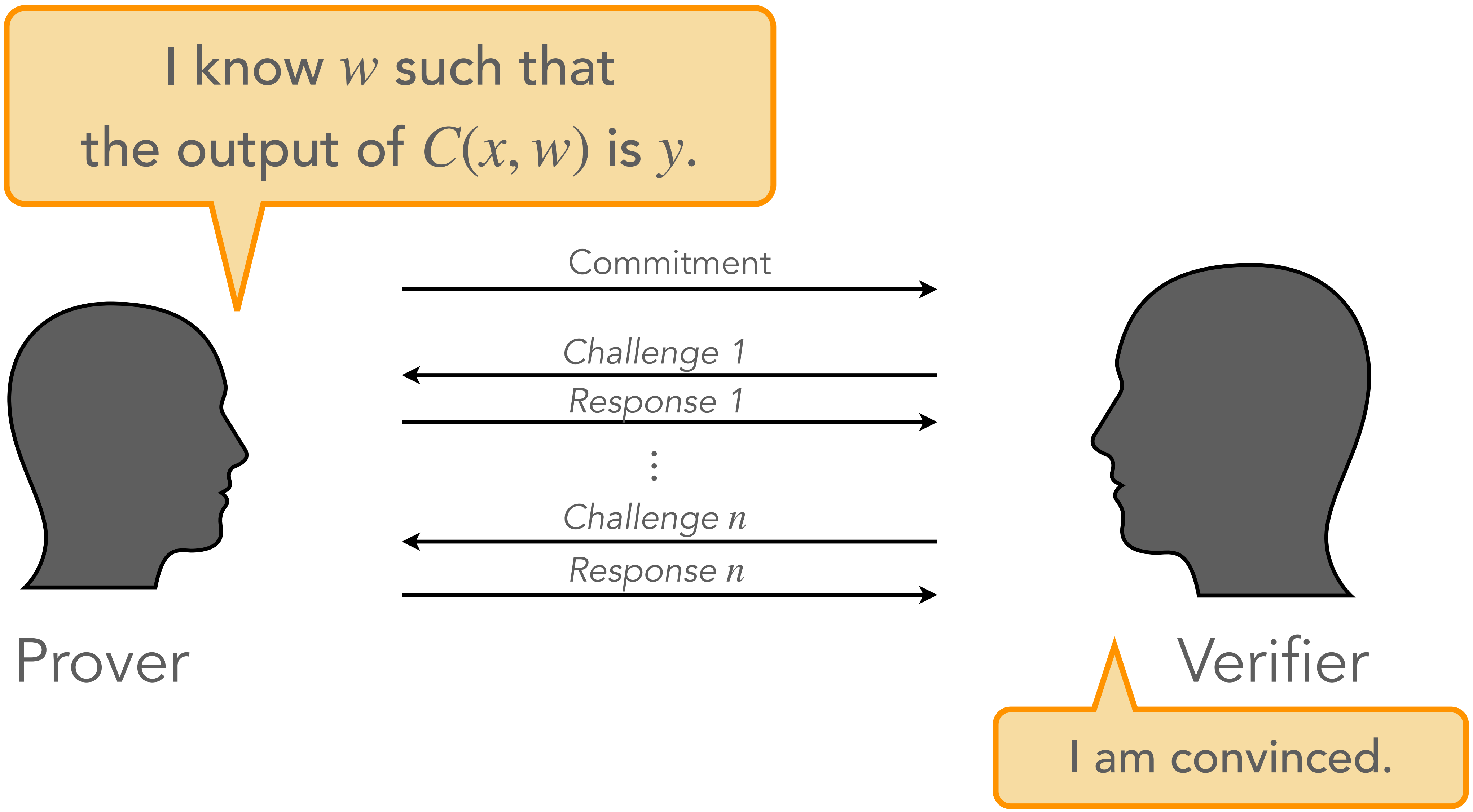
Efficiency: the communication between the prover and the verifier should ***be small***.

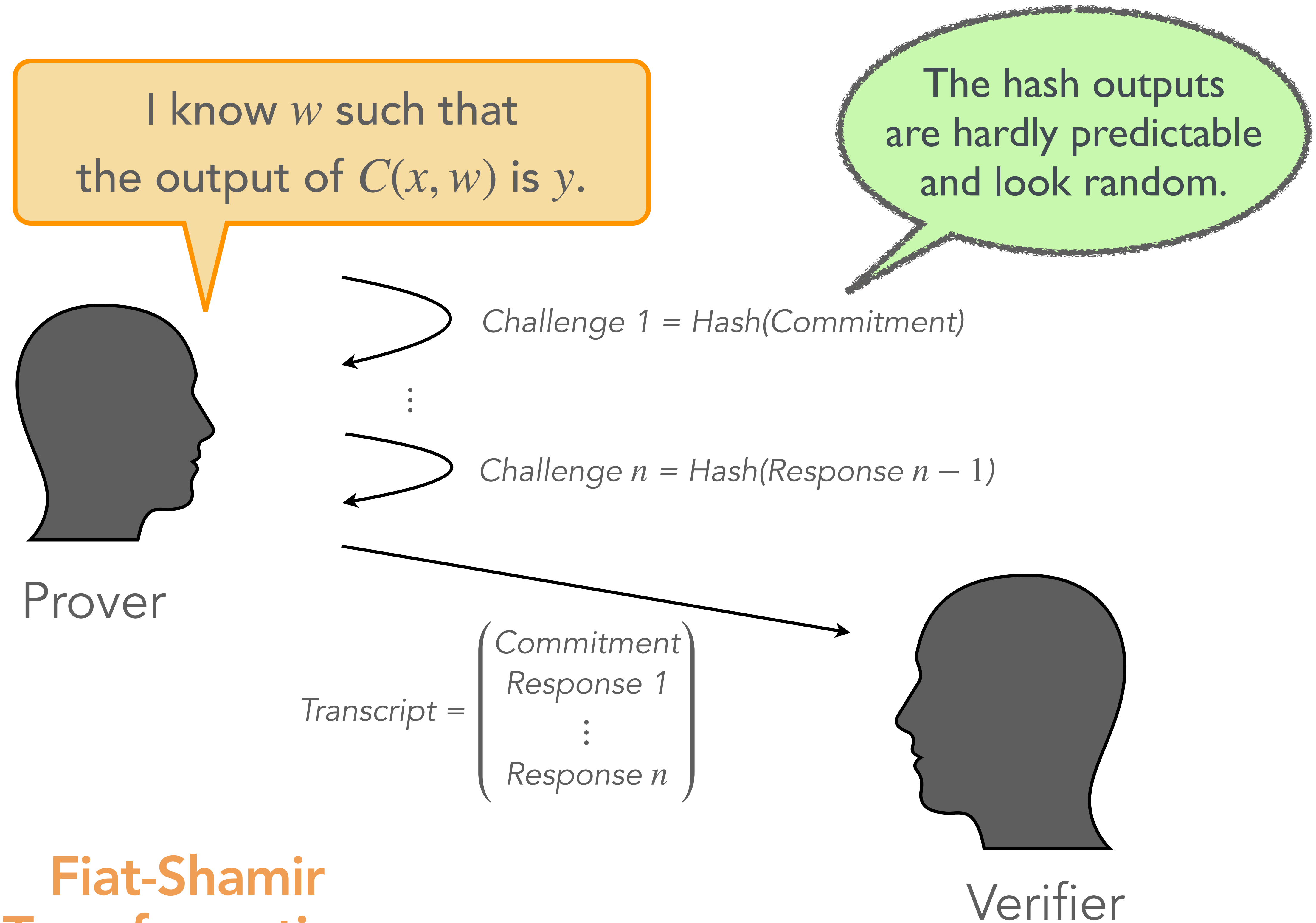
Proof Systems — Properties

- Completeness
- Soundness

Optional properties:

- Zero-Knowledge
- Efficient Verification
- Short communication


I know w such that
the output of $C(x, w)$ is y.
Commitment
Challenge 1
Response 1
Challenge n
Response n
Prover
Verifier
I am convinced.



Fiat-Shamir Transformation

Proof Systems — Properties

- Completeness
- Soundness

Optional properties:

- Zero-Knowledge
- Efficient Verification
- Short communication
- Non-interactive

Proof Systems — Properties

- Completeness
- Soundness

<u>Optional properties:</u>

- Zero-Knowledge
- Efficient Verification
- Short communication
- Non-interactive
- Quantum-resilient (i.e. post-quantum)

Proof Systems — Properties

- Completeness
- Soundness

Optional properties:

- Zero-Knowledge
- Efficient Verification
- Short communication
- Non-interactive

When both verification time and communication is very small compared to the size of C, we say that the proof system is

***succinct*.**

Proof Systems — Properties

- Completeness
- Soundness

Optional properties:

- Zero-Knowledge
- Efficient Verification
- Short communication
- Non-interactive

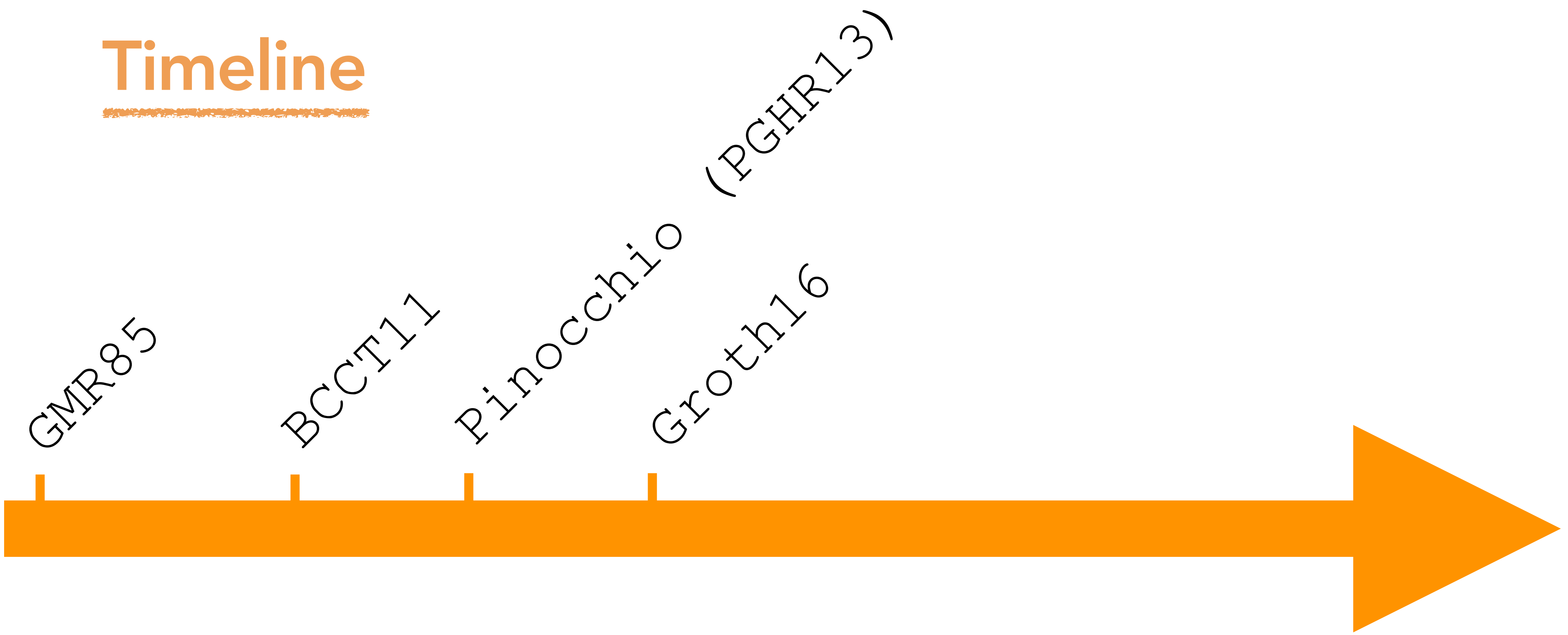
When both verification time and communication is very small compared to the size of C, we say that the proof system is

succinct.

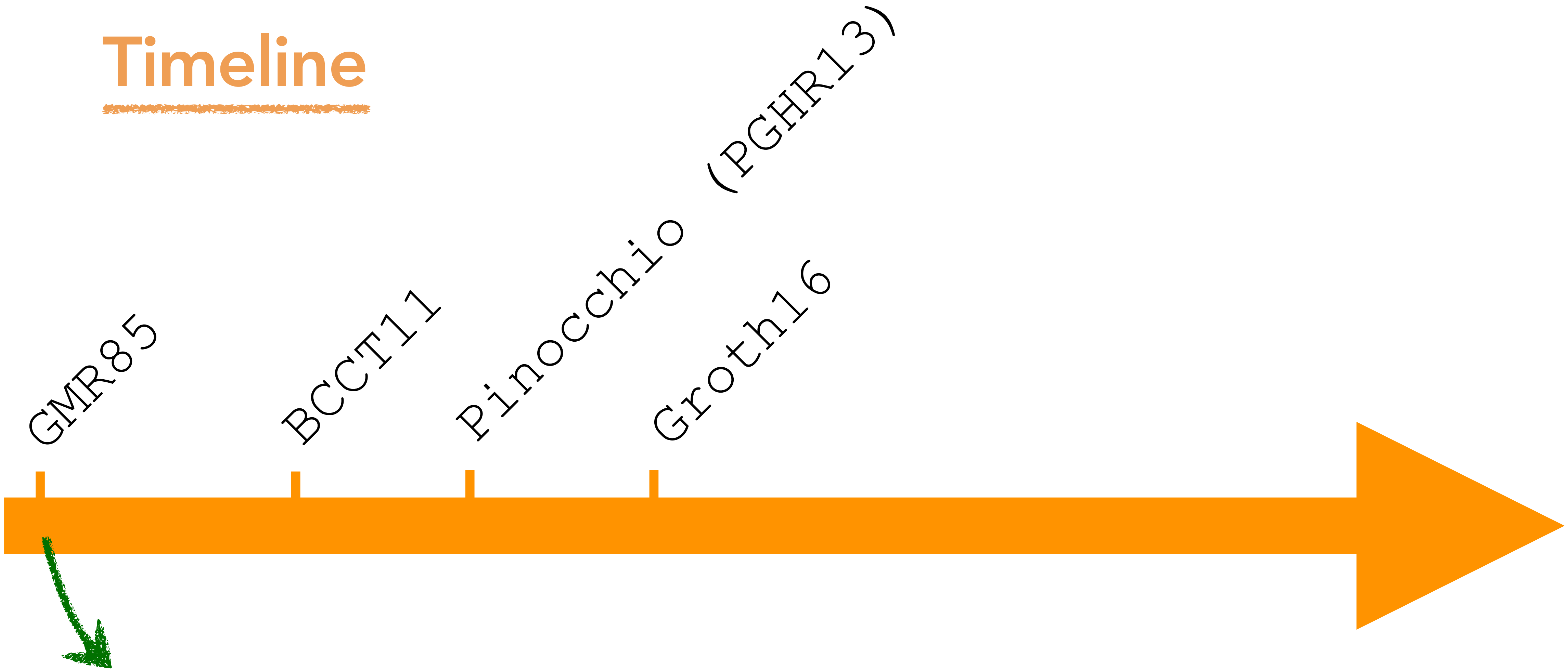
SNARK: Succinct Non-Interactive Arguments of Knowledge

State of the Art of the SNARK Technology

Timeline



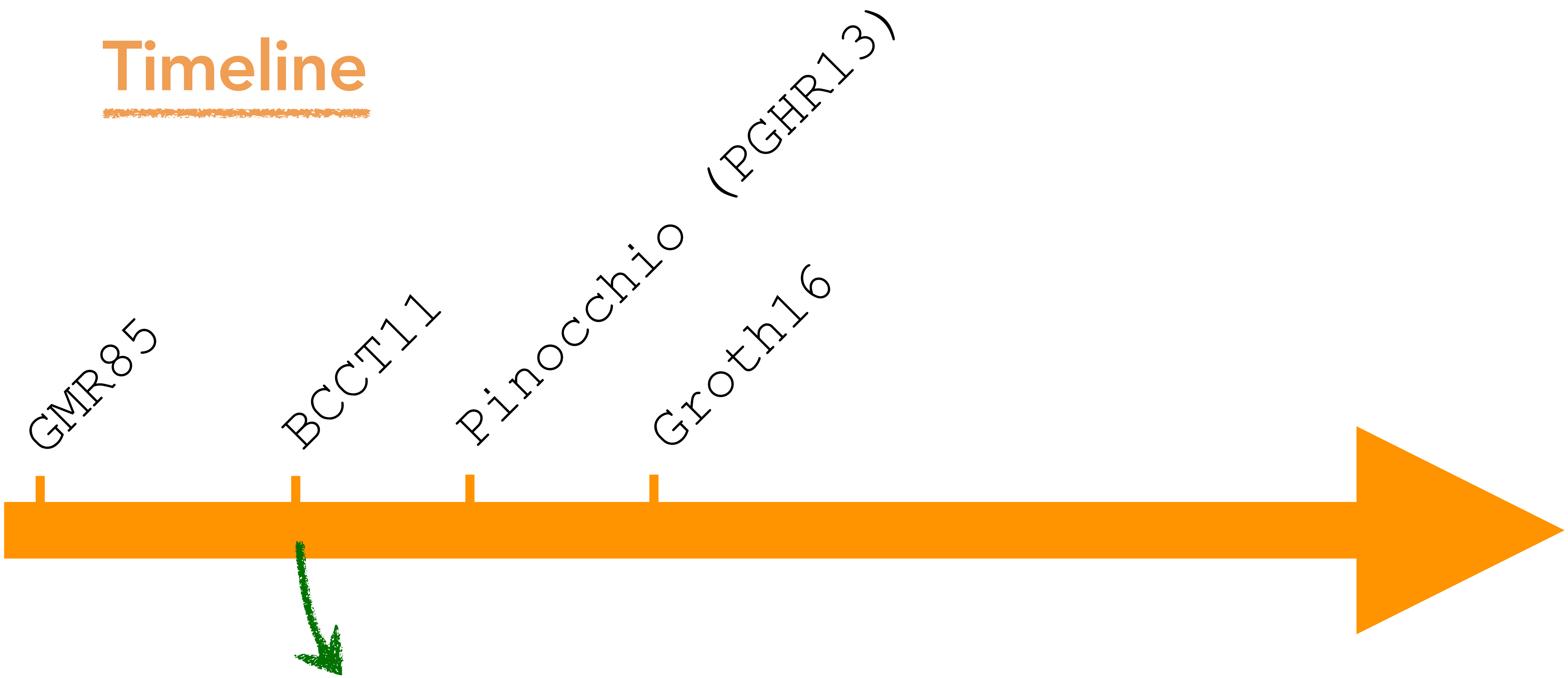
Timeline



[GMR85] *The Knowledge Complexity of Interactive Proof-Systems.*
Goldwasser, Micali, Rackoff. 1885

Seminal article introducing the zero-knowledge proofs

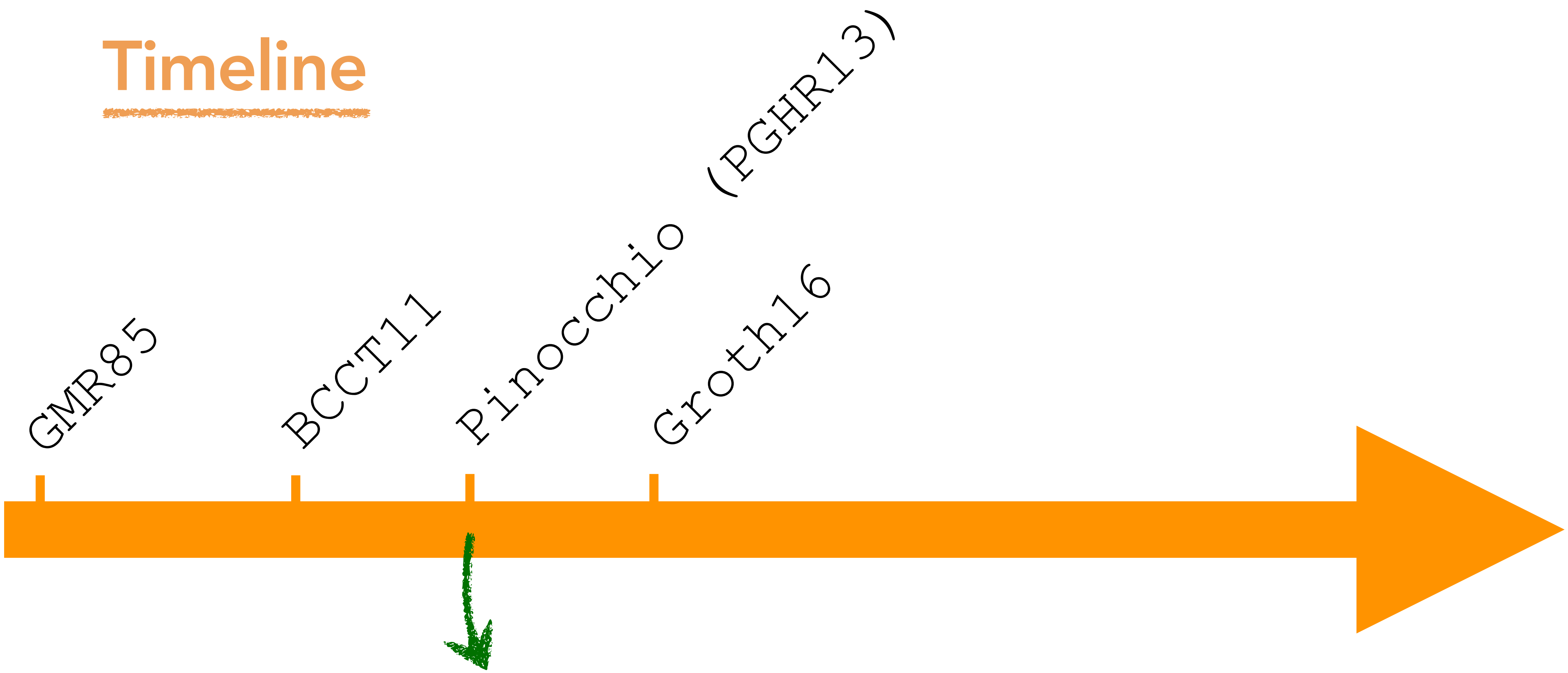
Timeline



[BCCT11] *From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again*. Bitansky, Canetti, Chiesa, Tromer. 2011

Article that introduces the notion of zk-SNARK.

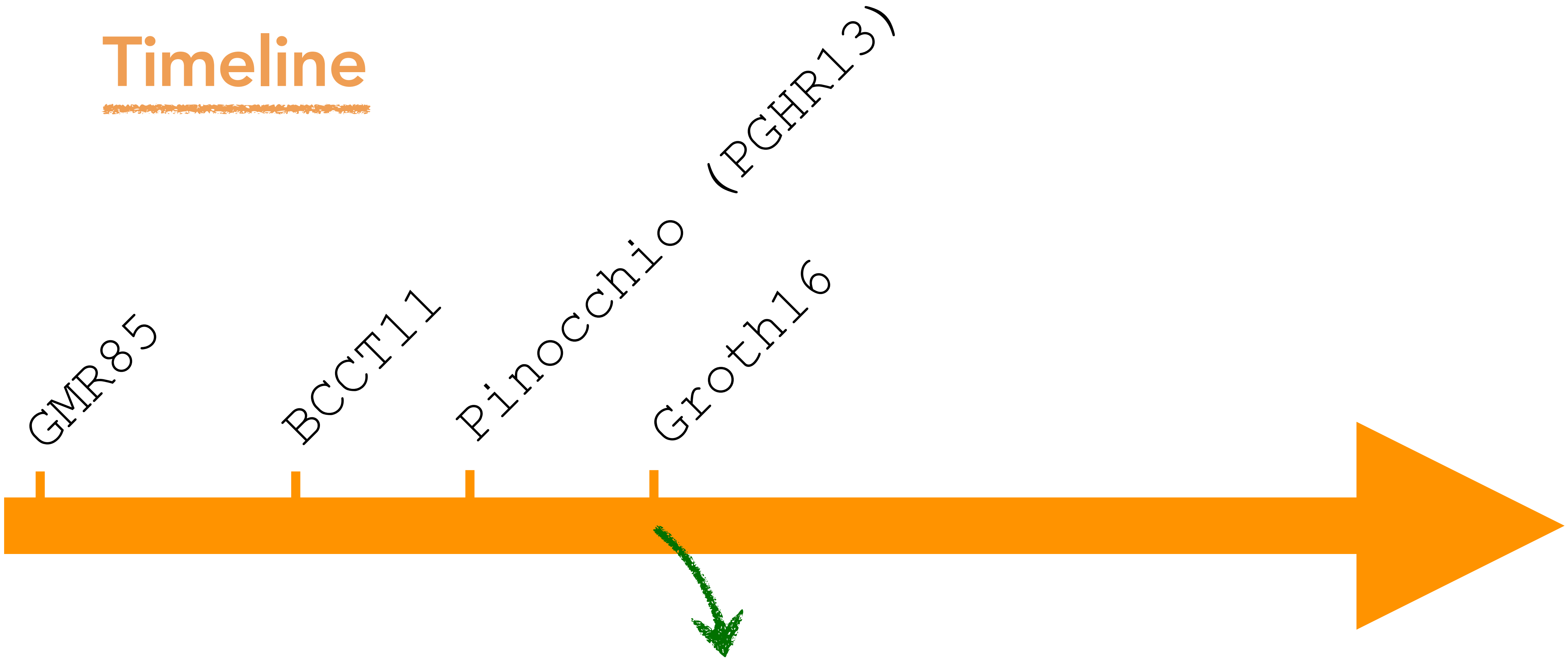
Timeline



[PGHR13] *Pinocchio: Nearly Practical Verifiable Computation*. Parno, Gentry, Howell, Raykova. 2013

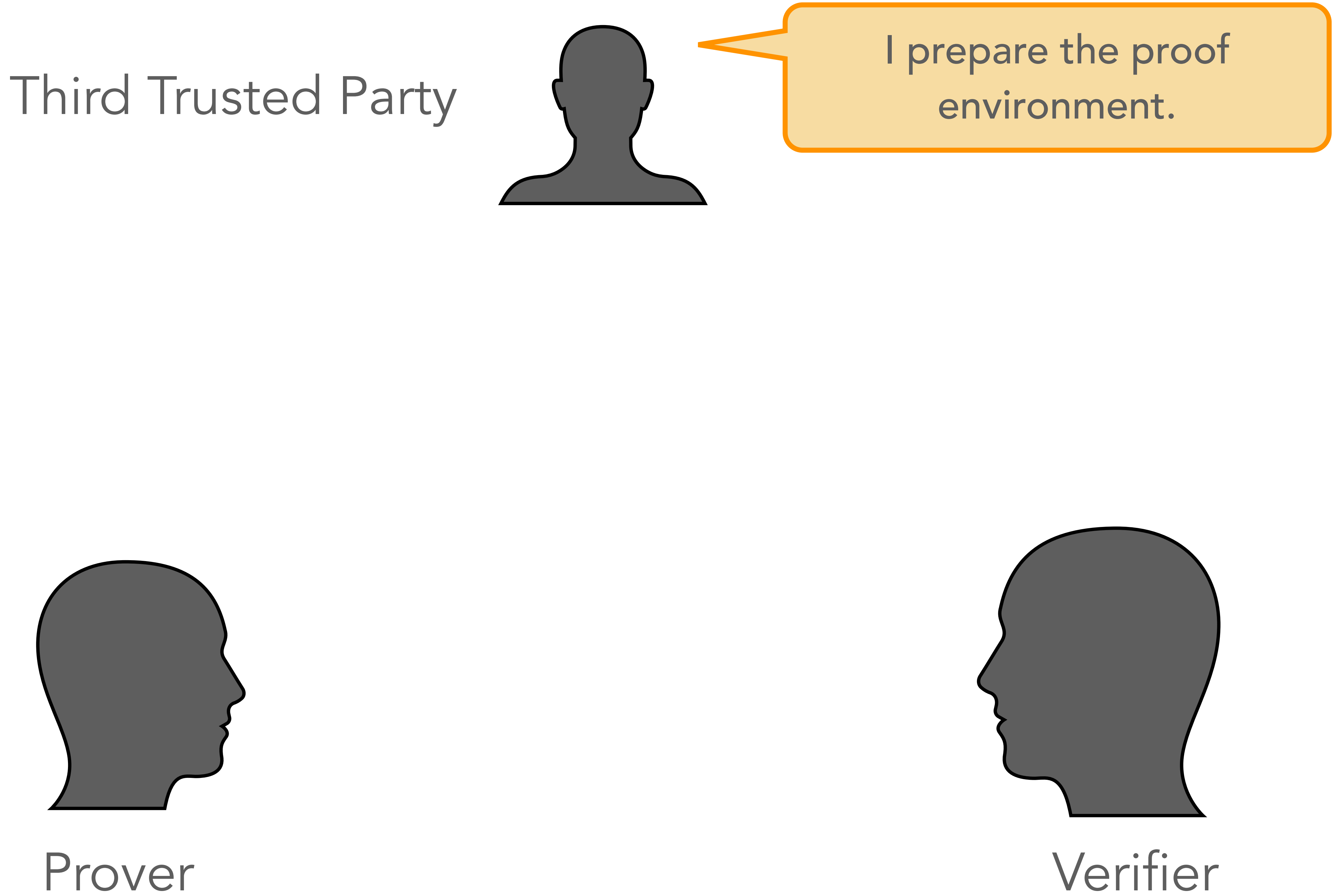
First practical SNARK for general computing.

Timeline



[Groth16] *On the Size of Pairing-based Non-interactive Arguments*. Groth. 2016

Highly efficient zk-SNARK (still very used).

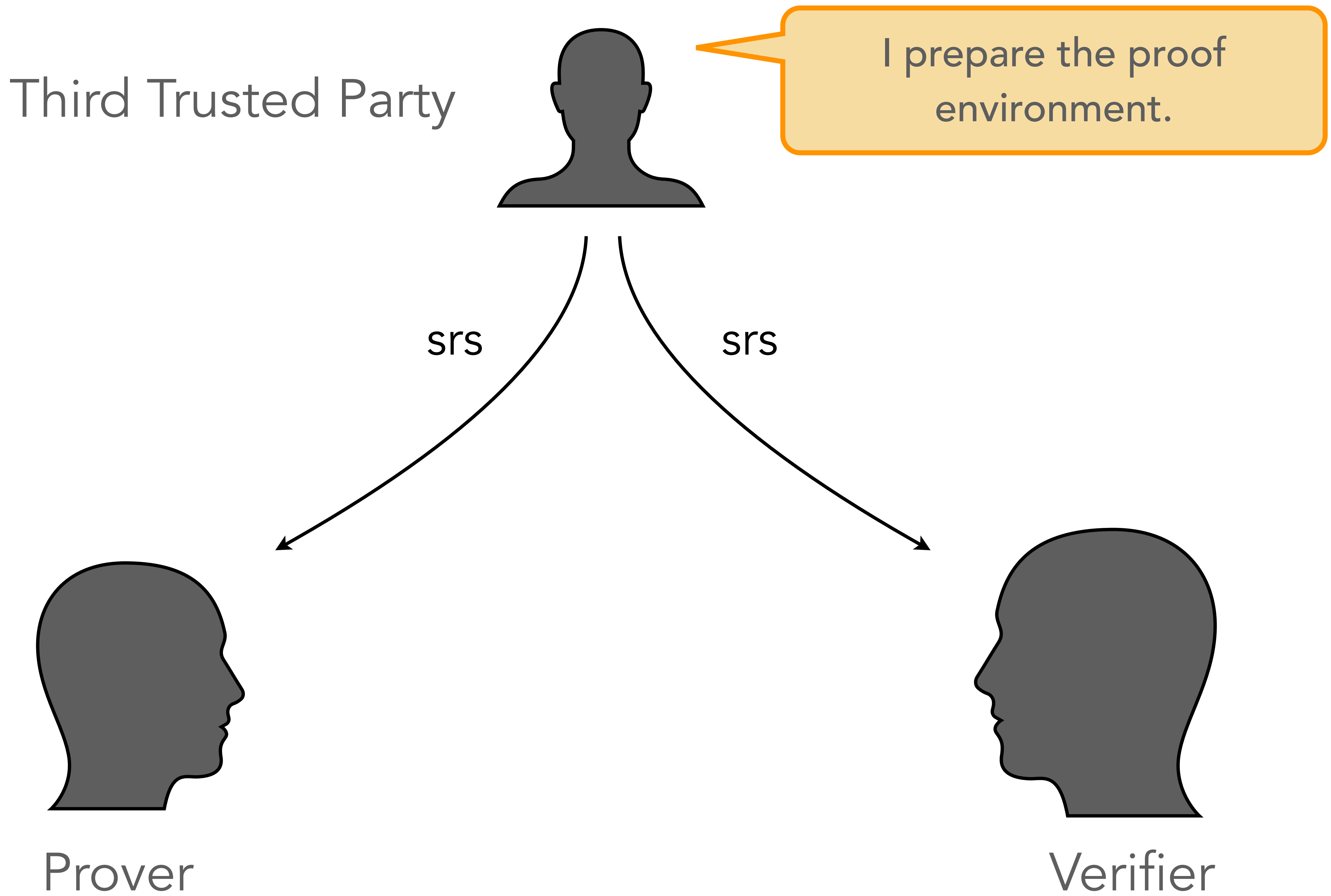

Third Trusted Party
I prepare the proof environment.
Prover
Verifier

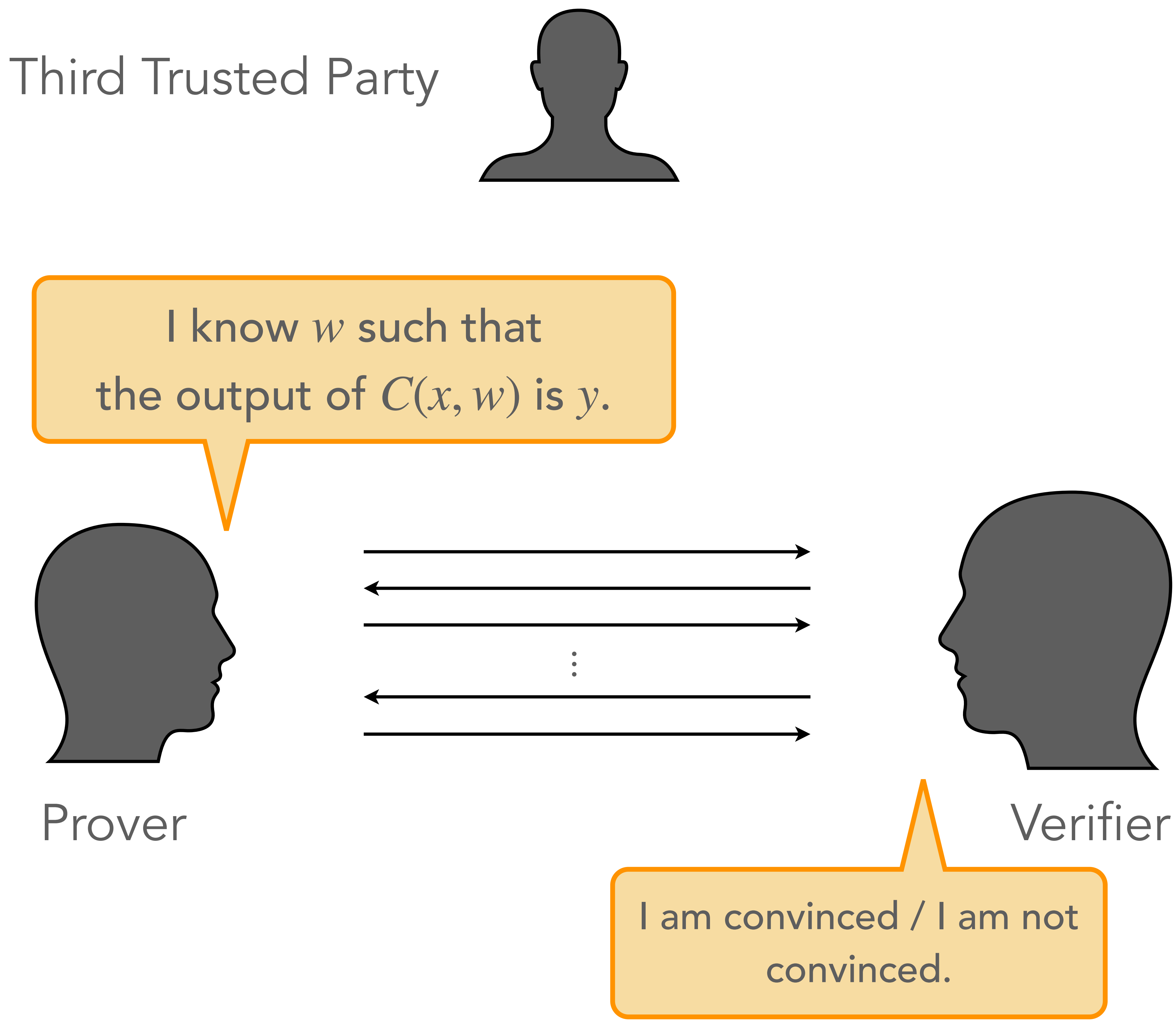
Third Trusted Party

- Sample a random value r.
- Generate a *structured reference string*

$$\mathsf{srs} \leftarrow \text{Procedure}(r)$$

- Discard the toxic waste r.


Third Trusted Party
I prepare the proof environment.
srs
srs
Prover
Verifier


Third Trusted Party
I know w such that the output of $C(x, w)$ is y.
Prover
Verifier
I am convinced / I am not convinced.

Dealing with Trusted Setup

- Trusted Setup by Circuit:

  the trusted set up works only
  for the considered circuit C.

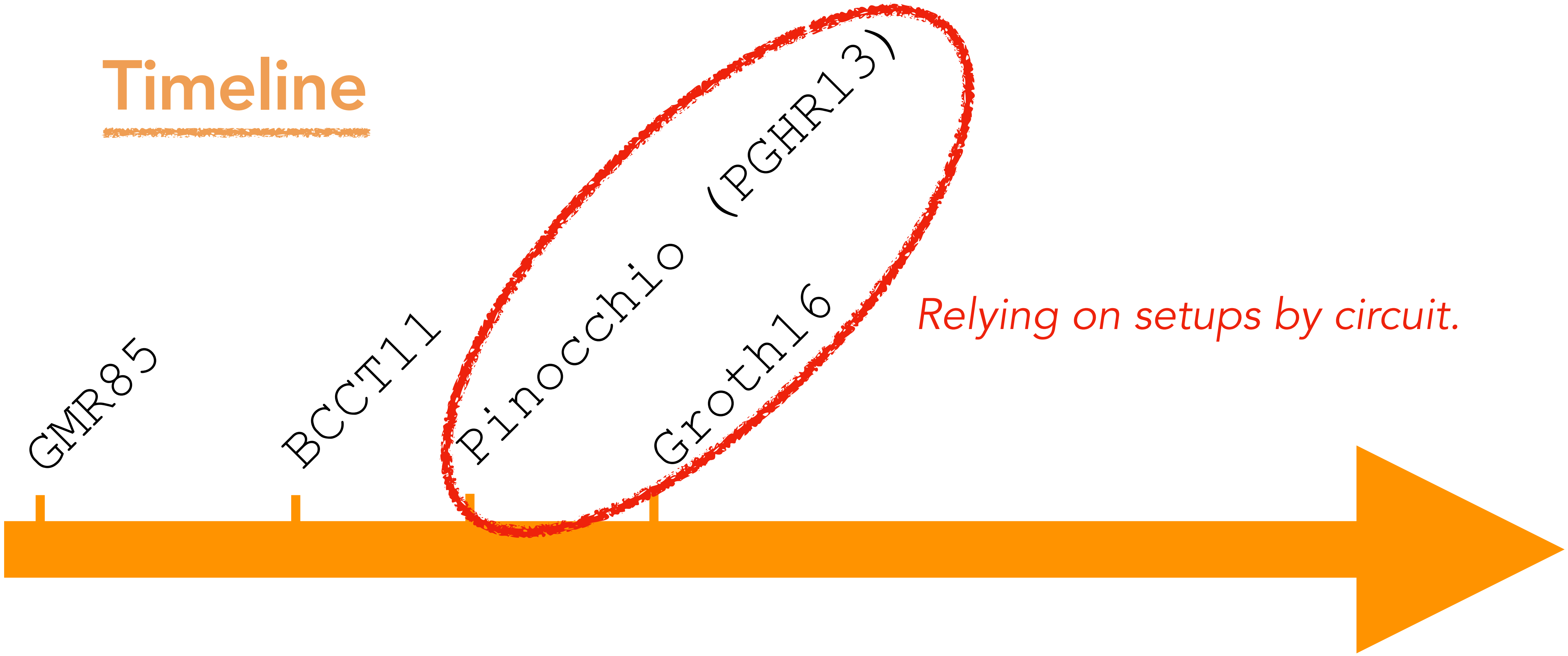
- Trusted Universal Setup:

  the trusted set up needs to be initiated
  only one (and will work for any circuit C).

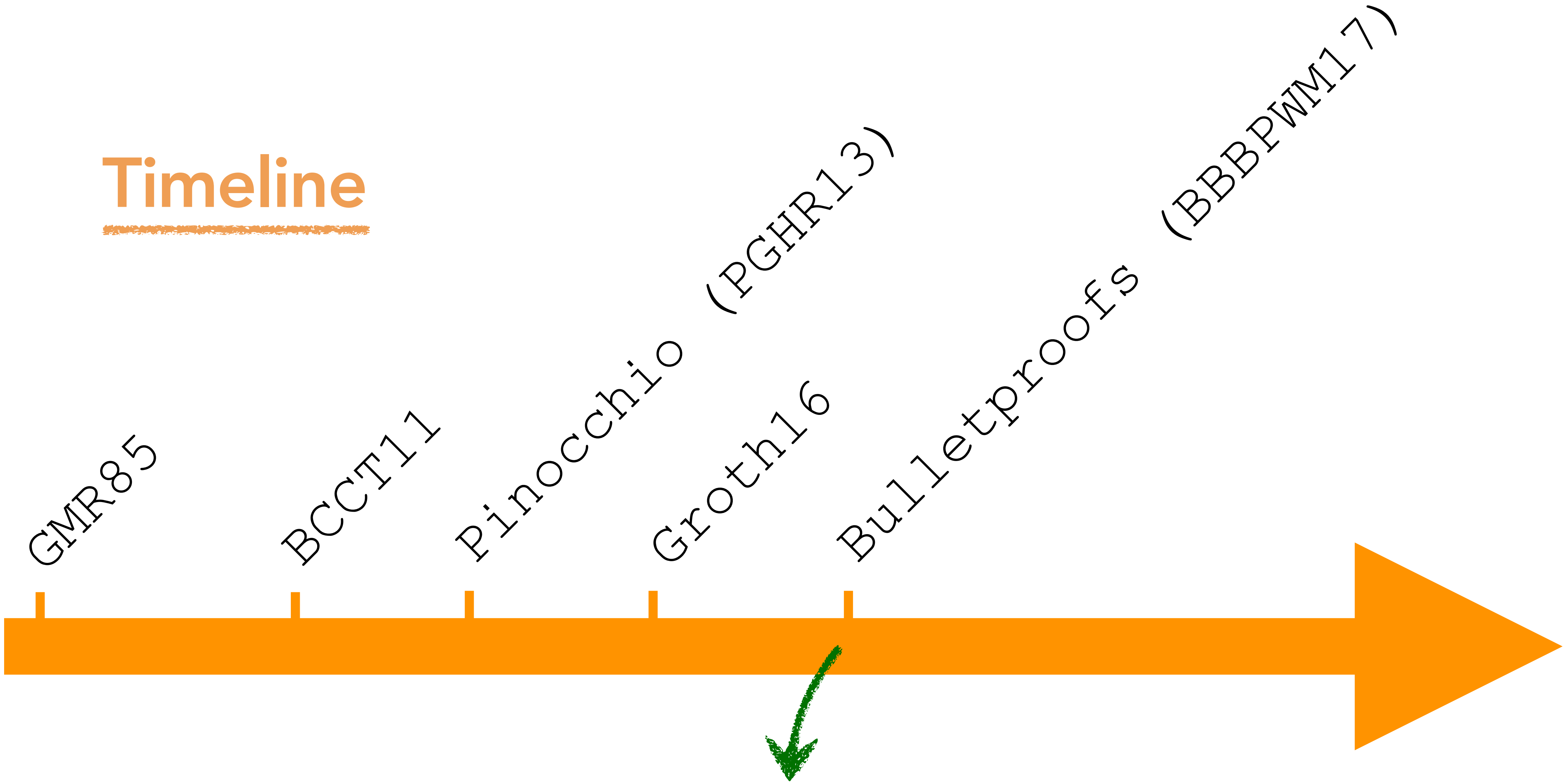
- Transparent Setup:

  no trusted set up is need.

Timeline


GMR85
BCCT11
Pinocchio (PGHR13)
Groth16
Relying on setups by circuit.

Timeline

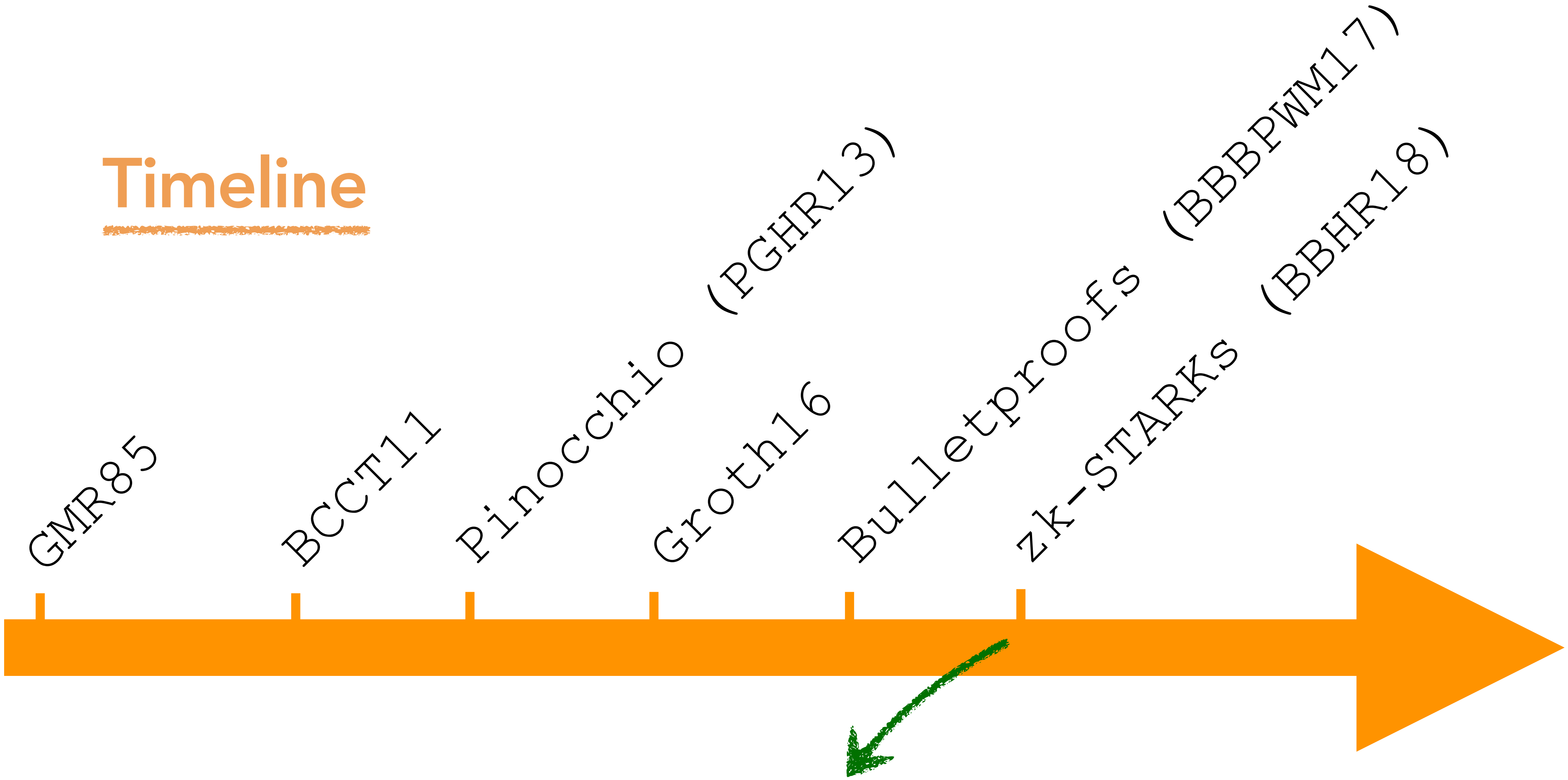


[BBBPWM17] *Bulletproofs: Short Proofs for Confidential Transactions and More*. Bünz, Bootle, Boneh, Poelstra, Wuille, Maxwell. 2017

One of the first efficient transparent proof systems

(no efficient verification)

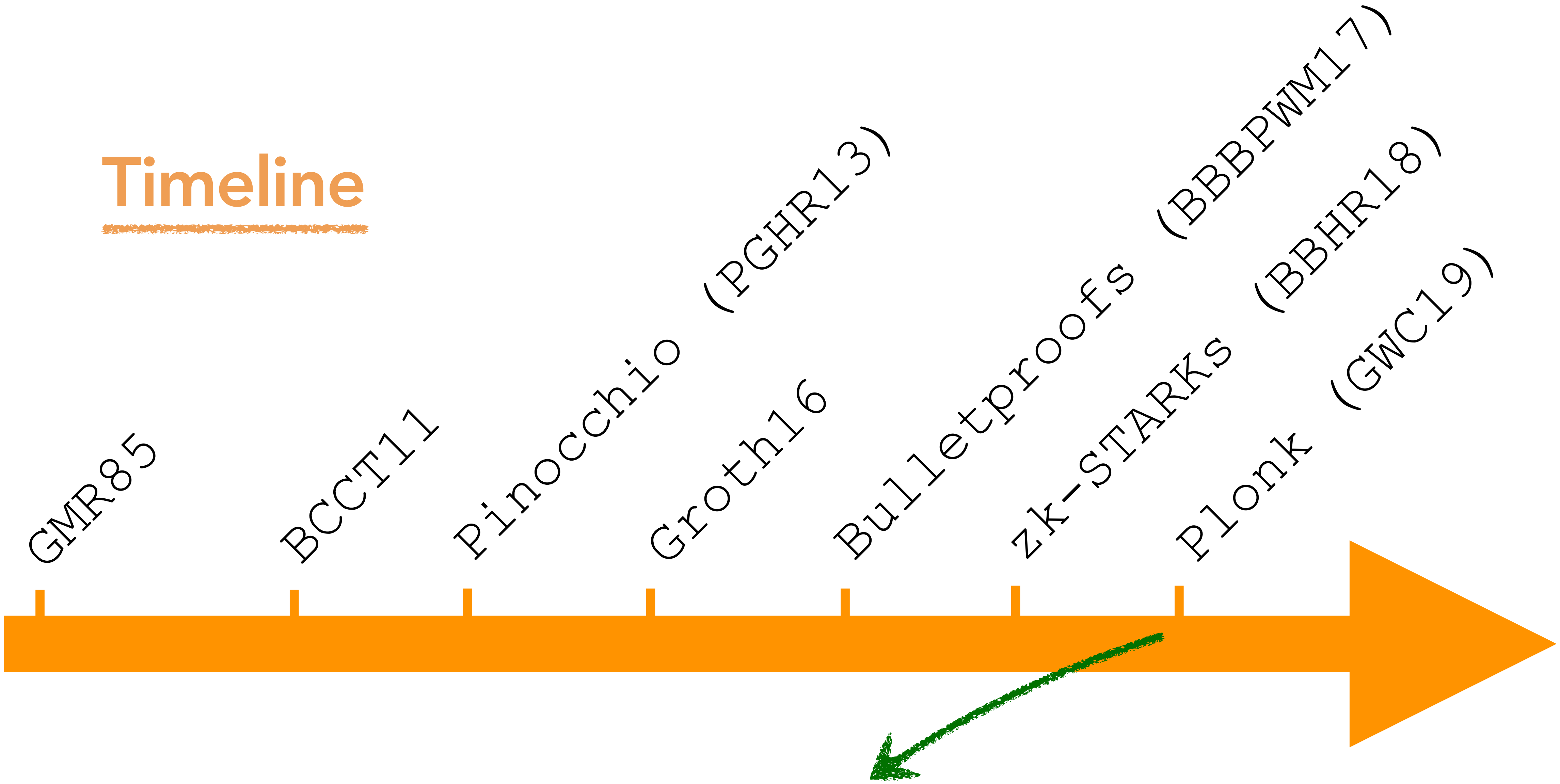
Timeline



[BBHR18] *Scalable, transparent, and post-quantum secure computational integrity*. Ben-Sasson, Bentov, Horesh, Riabzev. 2018

One of the first efficient <u>transparent</u> and <u>post-quantum</u> proof systems

Timeline



[GWC19] *Plonk: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Gabizon, Williamson, Ciobotaru. 2019

Practically-efficient proof system relying on universal setup

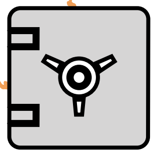
A (non-exhaustive) list of proof systems

ZKP System	Publication year	Protocol	Transparent	Universal	Plausibly Post-Quantum Secure	Programming Paradigm
Pinocchio[36]	2013	zk-SNARK	No	No	No	Procedural
Geppetto[37]	2015	zk-SNARK	No	No	No	Procedural
TinyRAM[38]	2013	zk-SNARK	No	No	No	Procedural
Buffet[39]	2015	zk-SNARK	No	No	No	Procedural
ZoKrates[40]	2018	zk-SNARK	No	No	No	Procedural
xJsnark[41]	2018	zk-SNARK	No	No	No	Procedural
vRAM[42]	2018	zk-SNARG	No	Yes	No	Assembly
vnTinyRAM[43]	2014	zk-SNARK	No	Yes	No	Procedural
MIRAGE[44]	2020	zk-SNARK	No	Yes	No	Arithmetic Circuits
Sonic[45]	2019	zk-SNARK	No	Yes	No	Arithmetic Circuits
Marlin[46]	2020	zk-SNARK	No	Yes	No	Arithmetic Circuits
PLONK[47]	2019	zk-SNARK	No	Yes	No	Arithmetic Circuits
SuperSonic[48]	2020	zk-SNARK	Yes	Yes	No	Arithmetic Circuits
Bulletproofs[24]	2018	Bulletproofs	Yes	Yes	No	Arithmetic Circuits
Hyrax[49]	2018	zk-SNARK	Yes	Yes	No	Arithmetic Circuits
Halo[50]	2019	zk-SNARK	Yes	Yes	No	Arithmetic Circuits
Virgo[51]	2020	zk-SNARK	Yes	Yes	Yes	Arithmetic Circuits
Ligero[52]	2017	zk-SNARK	Yes	Yes	Yes	Arithmetic Circuits
Aurora[53]	2019	zk-SNARK	Yes	Yes	Yes	Arithmetic Circuits
zk-STARK[54]	2019	zk-STARK	Yes	Yes	Yes	Assembly
Zilch[35]	2021	zk-STARK	Yes	Yes	Yes	Object-Oriented

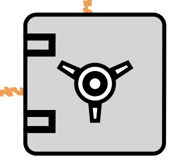
Source: Wikipedia (page "Zero-knowledge Proof")

A general SNARK framework

SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



A general SNARK framework

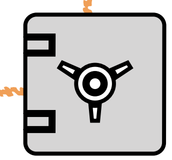
SNARK = Functional commitment schemes  + Interactive oracle proof (IOP)

- <u>Commitment scheme</u>: commit a *hidden value* that we can reveal later.

Too simple to build efficient SNARK system.

A general SNARK framework

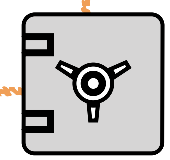
SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



- Commitment scheme: commit a *hidden value* that we can reveal later.
- Functional commitment scheme: commit a *hidden function* f for which we can reveal some $\{f(x_i)\}_i$ later.

A general SNARK framework

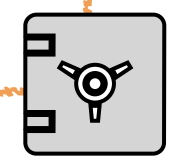
SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



Algorithm that describe how can get an efficient proof system from a generic functional commitment scheme (i.e. oracle)

A general SNARK framework

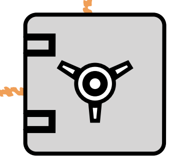
SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



When there is a trusted setup, it is usually because of the functional commitment scheme.

A general SNARK framework

SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



Phase 1/2: Committing procedure

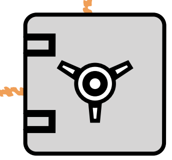
- We fix a wet of functions $\mathscr{F} = \{f : X \to Y\}$.
- The prover commits a given function $f \in \mathscr{F}$:

$$com_f \leftarrow \mathrm{Com}^{\rho}(f).$$

- The prover sends the commitment com_f to the verifier.

A general SNARK framework

SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



Commitment schemes usually rely on some randomness ρ to hide the committed data.

Phase 1/2: Committing procedure

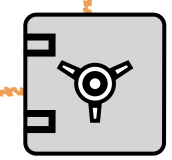
- We fix a wet of functions $\mathscr{F} = \{f : X \to Y\}$.
- The prover commits a given function $f \in \mathscr{F}$:

$$com_f \leftarrow \mathrm{Com}^{\rho}(f).$$

- The prover sends the commitment com_f to the verifier.

A general SNARK framework

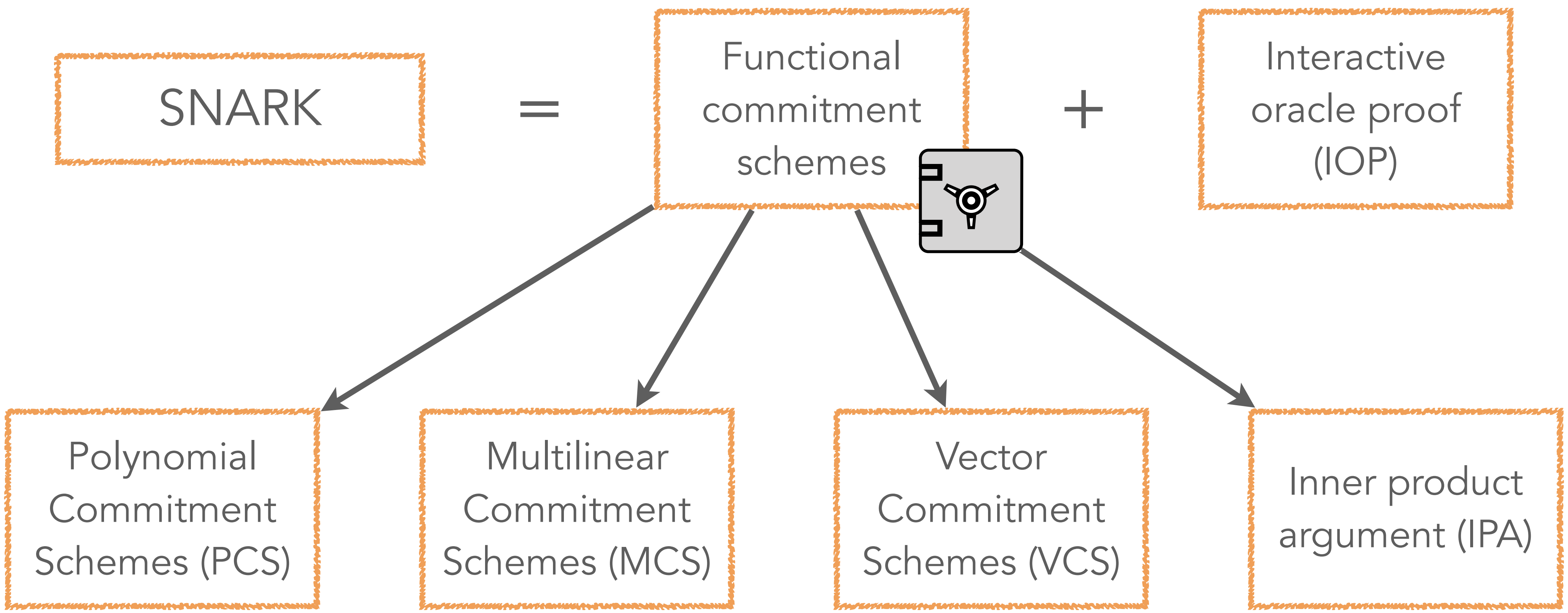
SNARK = Functional commitment schemes + Interactive oracle proof (IOP)



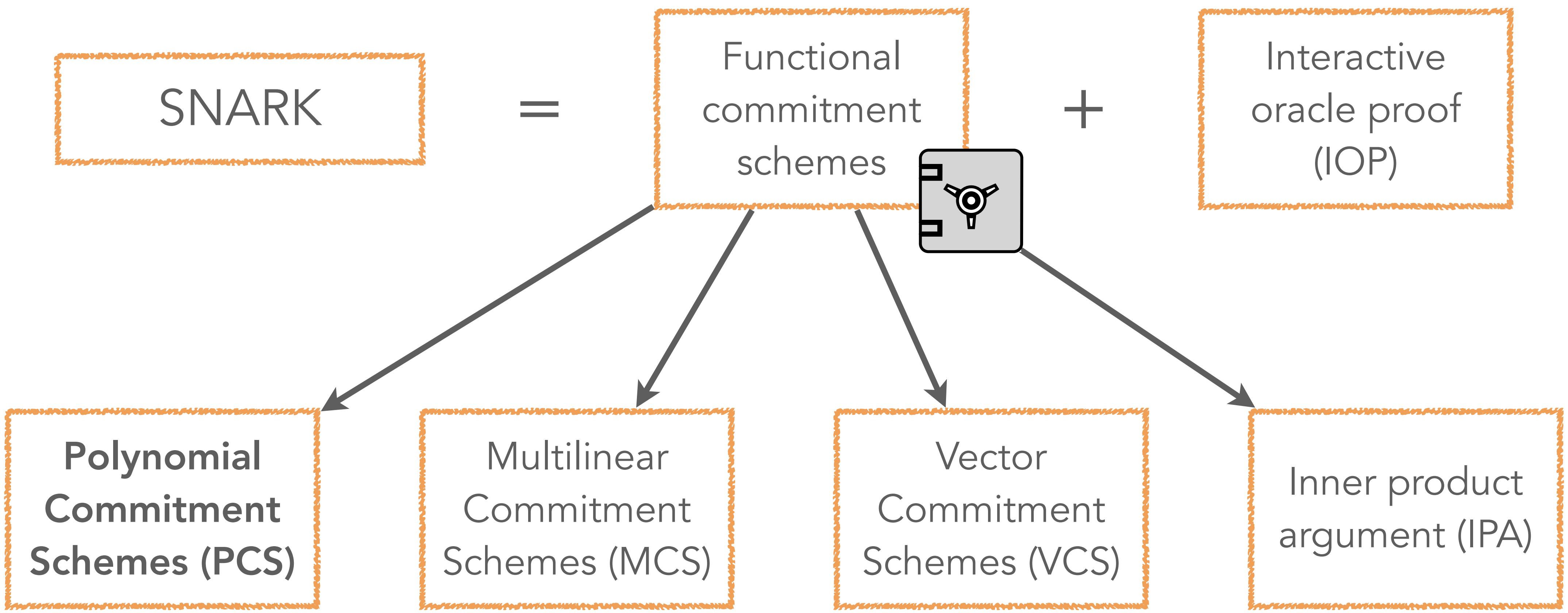
Phase 2/2: Verifying procedure

- The verifier sends some $x \in X$ to the prover.
- The prover replies with proof π together with some $y \in Y$.
- The verifier uses the proof π to verify that $f(x) = y$ and $f \in \mathscr{F}$.
- The verifier accept the claim iff the proof π is valid.

A general SNARK framework



A general SNARK framework

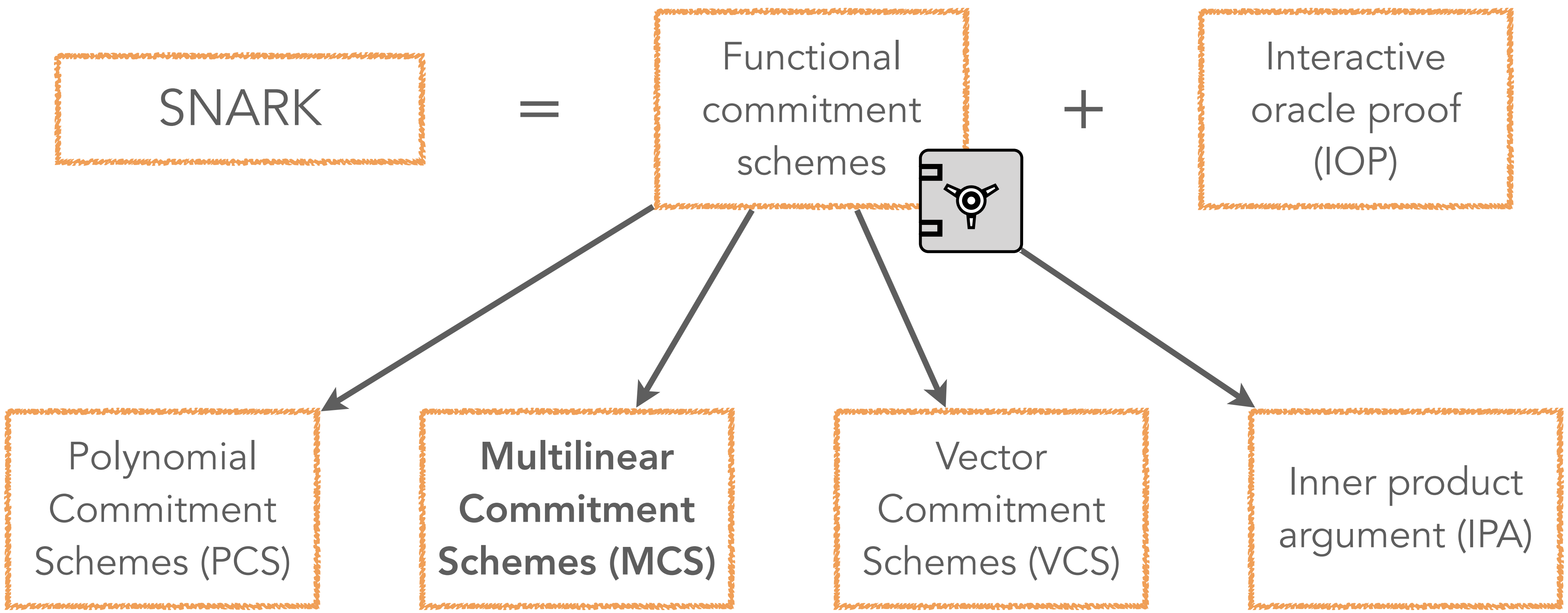


The set $\mathscr{F}$ is all the univariate polynomials of degree less or equal to d in a specific field:

$$\mathscr{F} = \mathbb{F}^{\leq d}[X]$$

KZG and FRI are famous polynomial commitment schemes.

A general SNARK framework



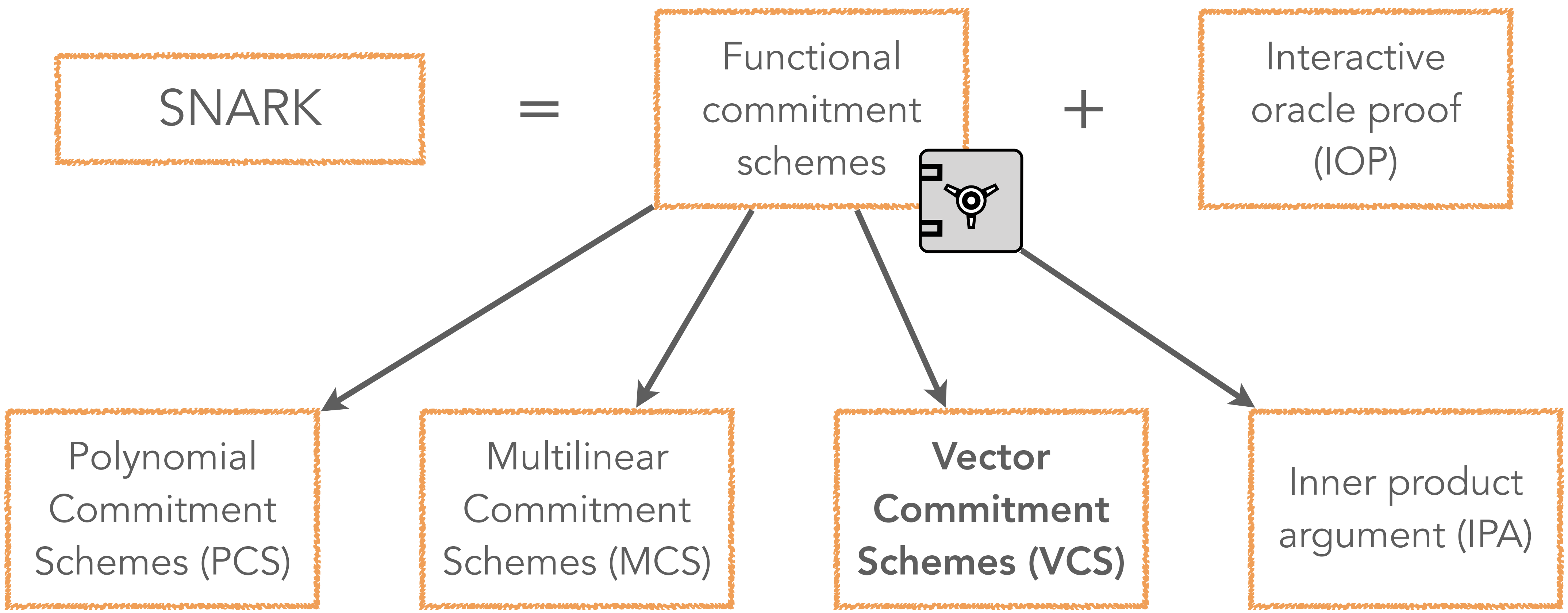
The set $\mathscr{F}$ is all the multivariate polynomials of degree at most 1:

$$\mathscr{F} = \mathbb{F}^{\leq 1}[X_1, \ldots, X_n]$$

For example:

$$f_1(X_1, X_2, X_3) = 5 + 3X_2 + X_3 \qquad \textit{or} \qquad f_2(X_1, X_2, X_3) = X_1 + X_2 + X_3$$

A general SNARK framework

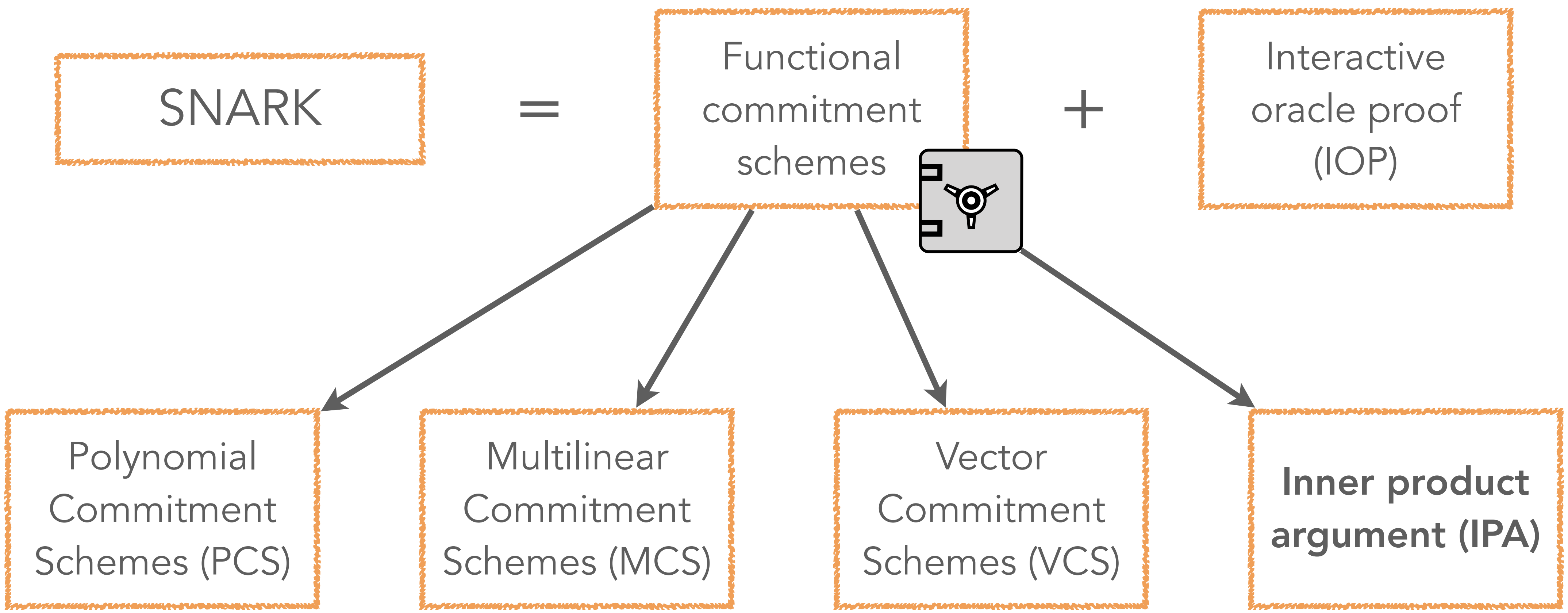


The set $\mathscr{F}$ is all the functions that are represented by vectors: we commit to a vector a dimension d.

$$\mathscr{F} = \{f_{\vec{u}} : i \mapsto u_i, \vec{u} := (u_1, \ldots, u_d)\}$$

Such a commitment scheme simply commits to a vector and enables us to reveal only few coordinates (and not the entire vector).

A general SNARK framework

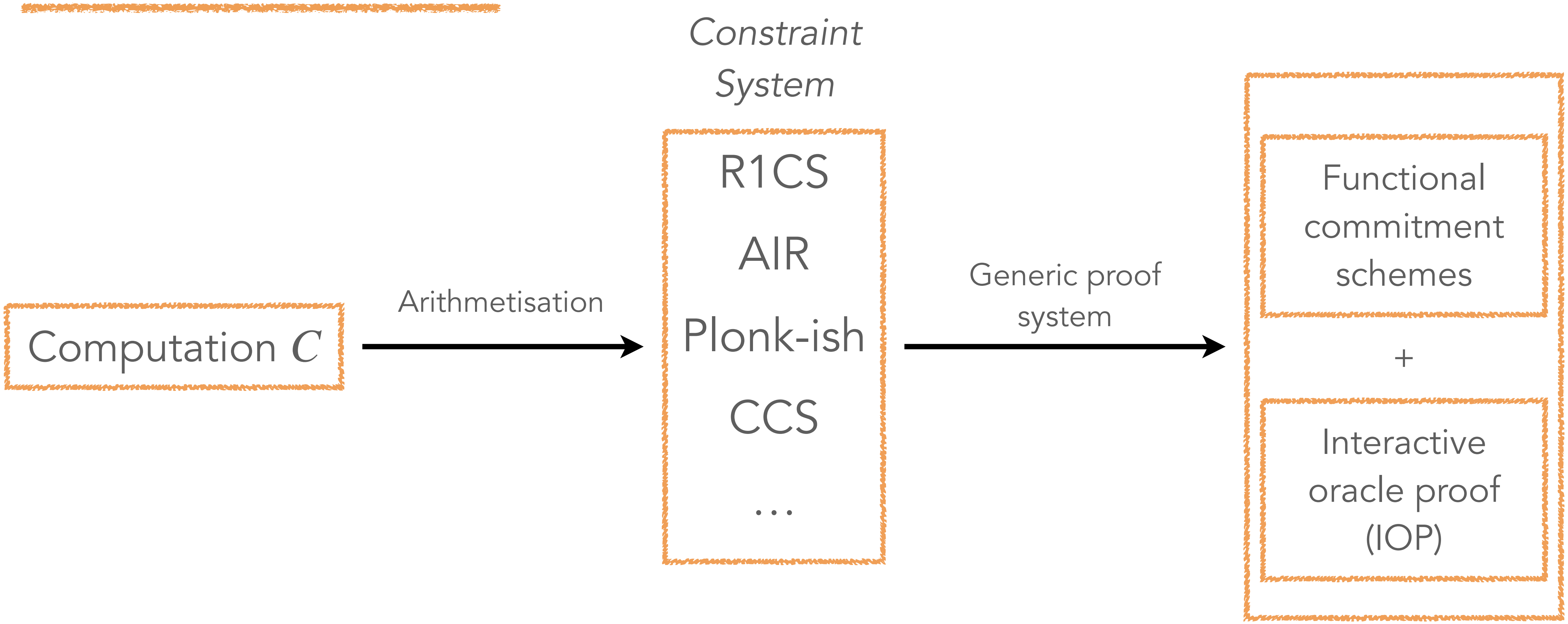


The set $\mathscr{F}$ is all the functions that are represented by vectors: we commit to a vector a dimension d.

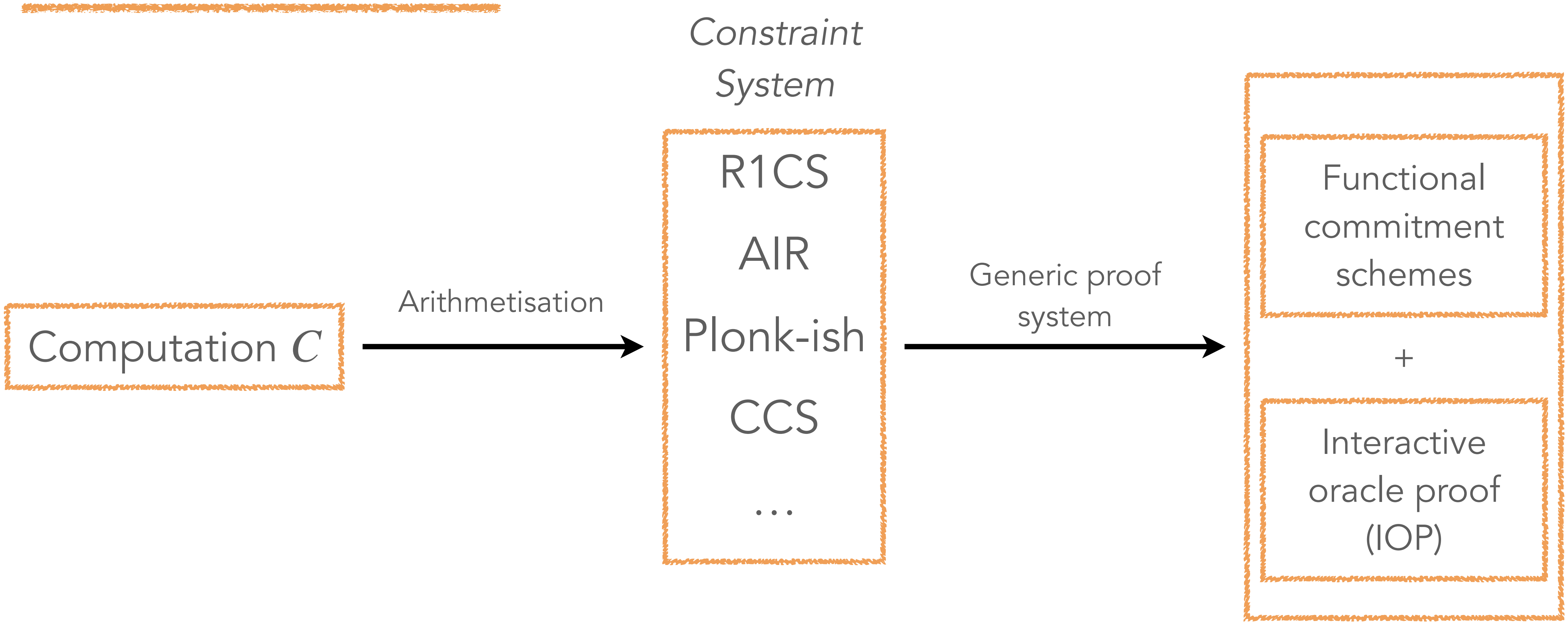
$$\mathscr{F} = \{f_{\vec{u}} : \vec{v} \mapsto \langle \vec{u}, \vec{v} \rangle\}$$

Such a commitment scheme simply commits to a vector and enables us to reveal only linear combinations of the coordinates.

Arithmetisation



Arithmetisation



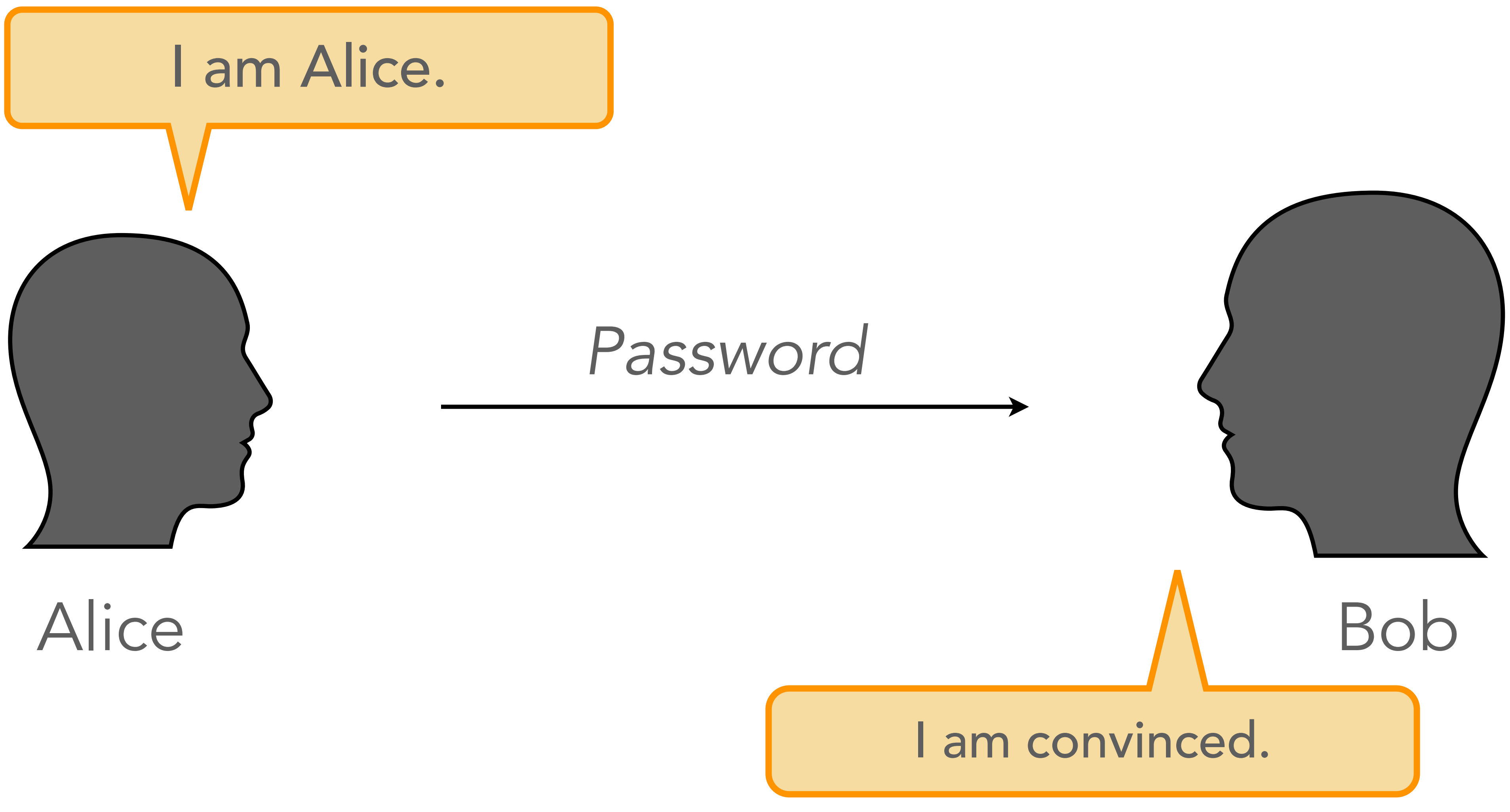
R1CS (Rank-1 Constraint Systems):

I know w such that the output of $C(x, w)$ is y.

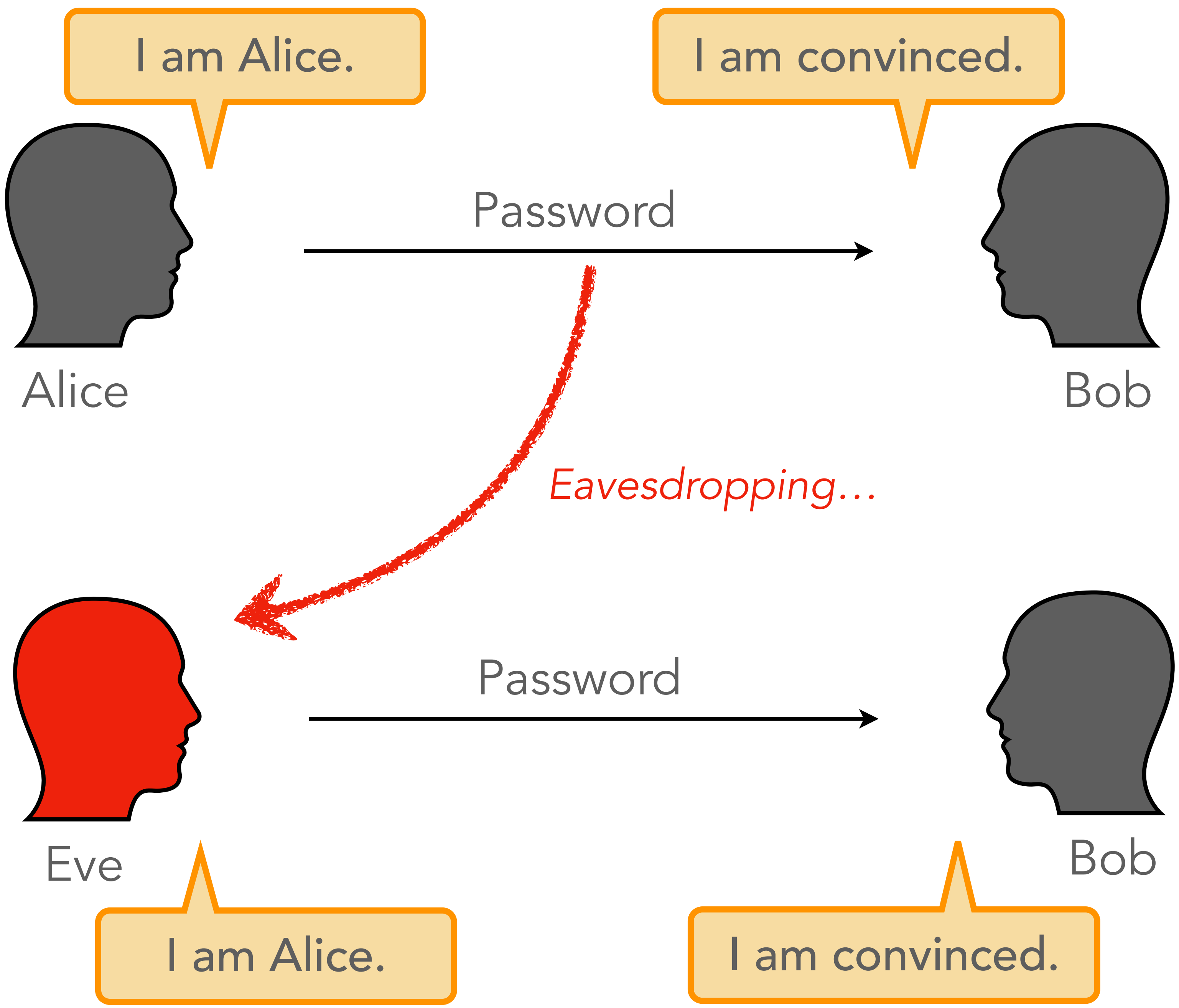
$\implies$ I know $\bar{w}$ such that $(A\bar{w}) \circ (B\bar{w}) = C\bar{w}$.

Applications

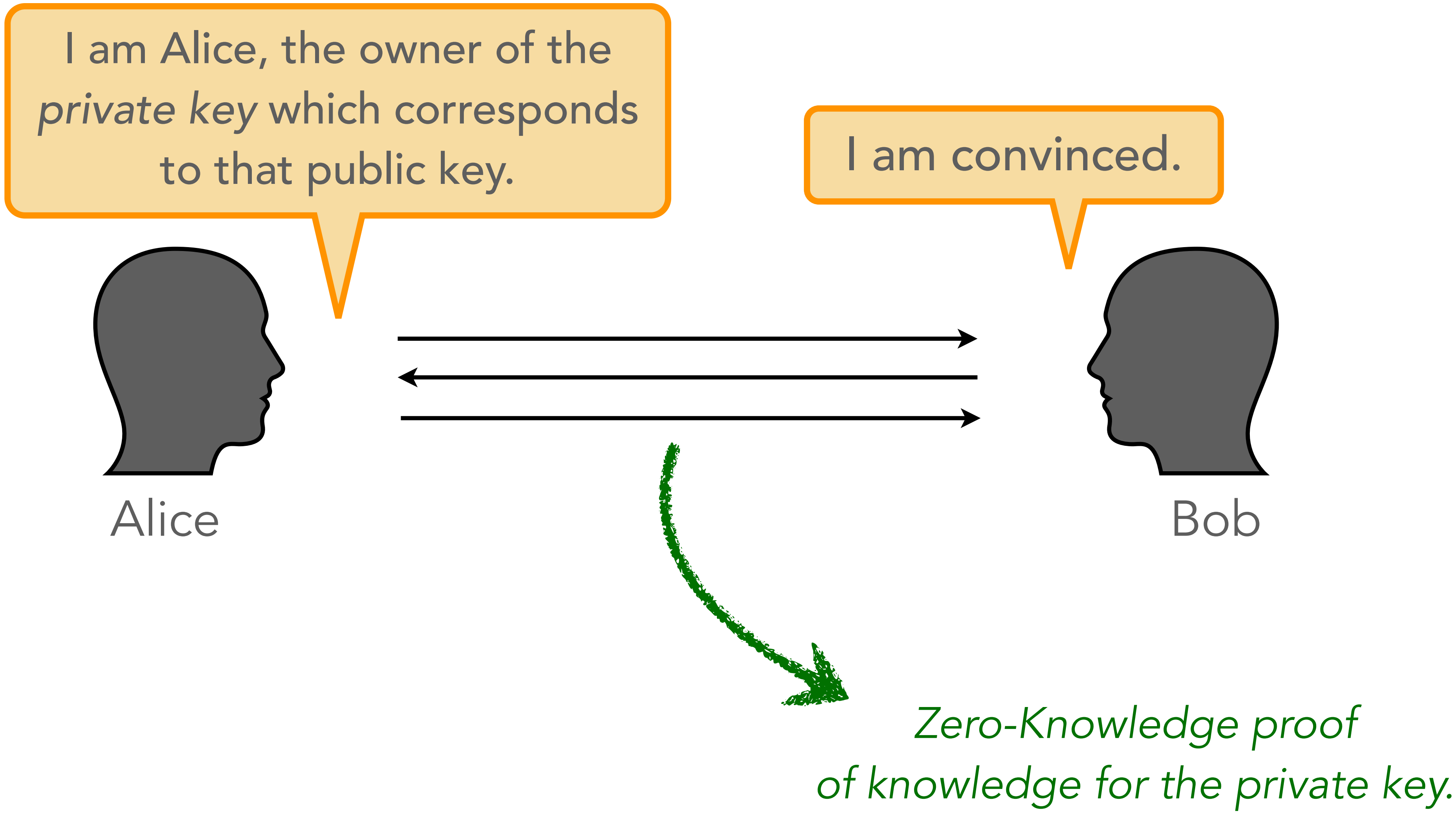
Authentication



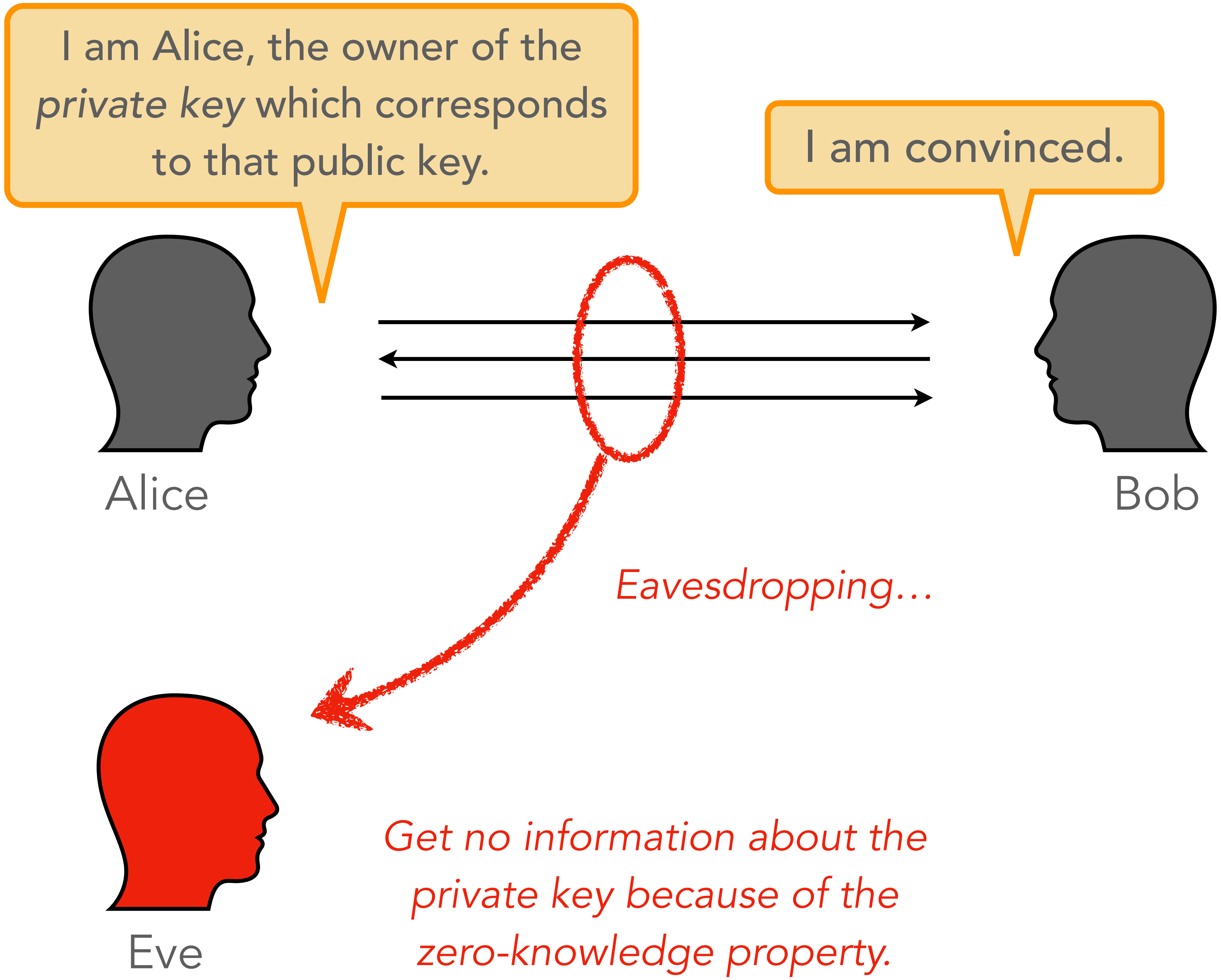
Authentication



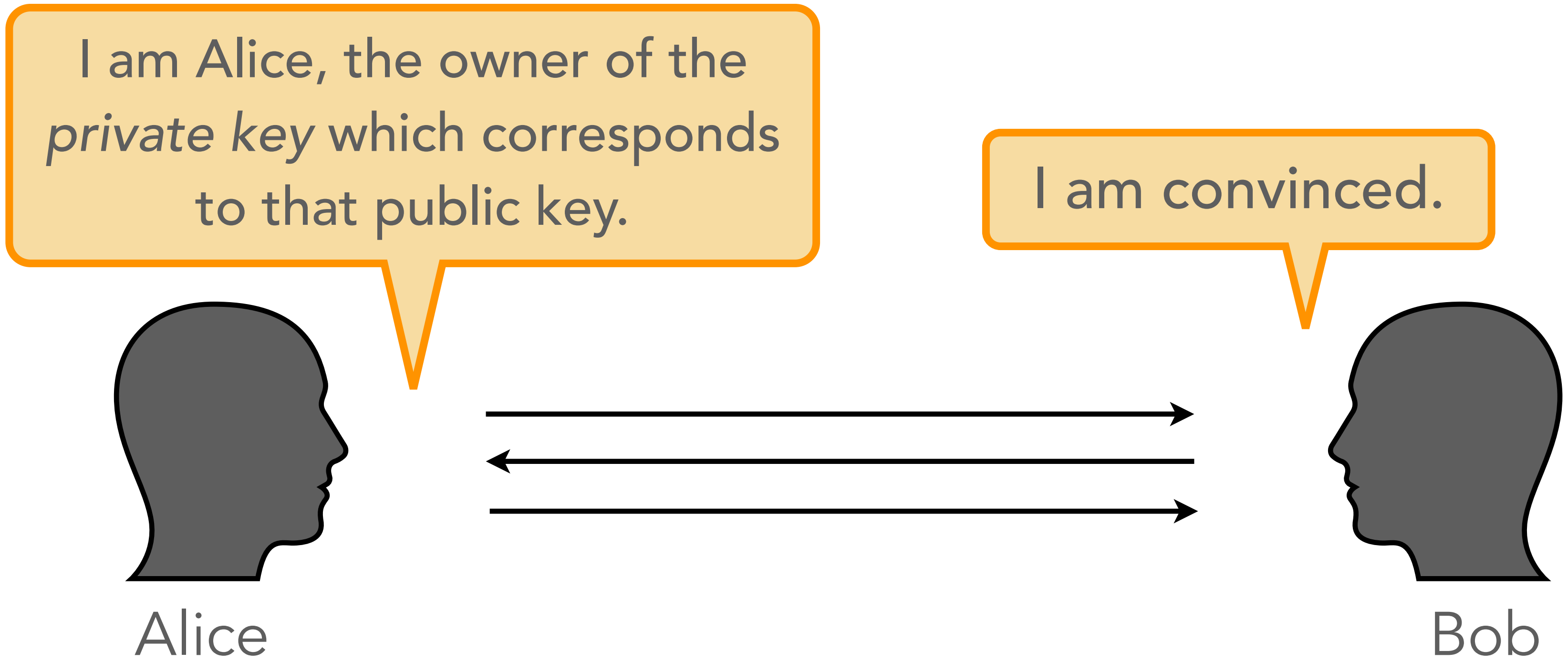
Authentication



Authentication

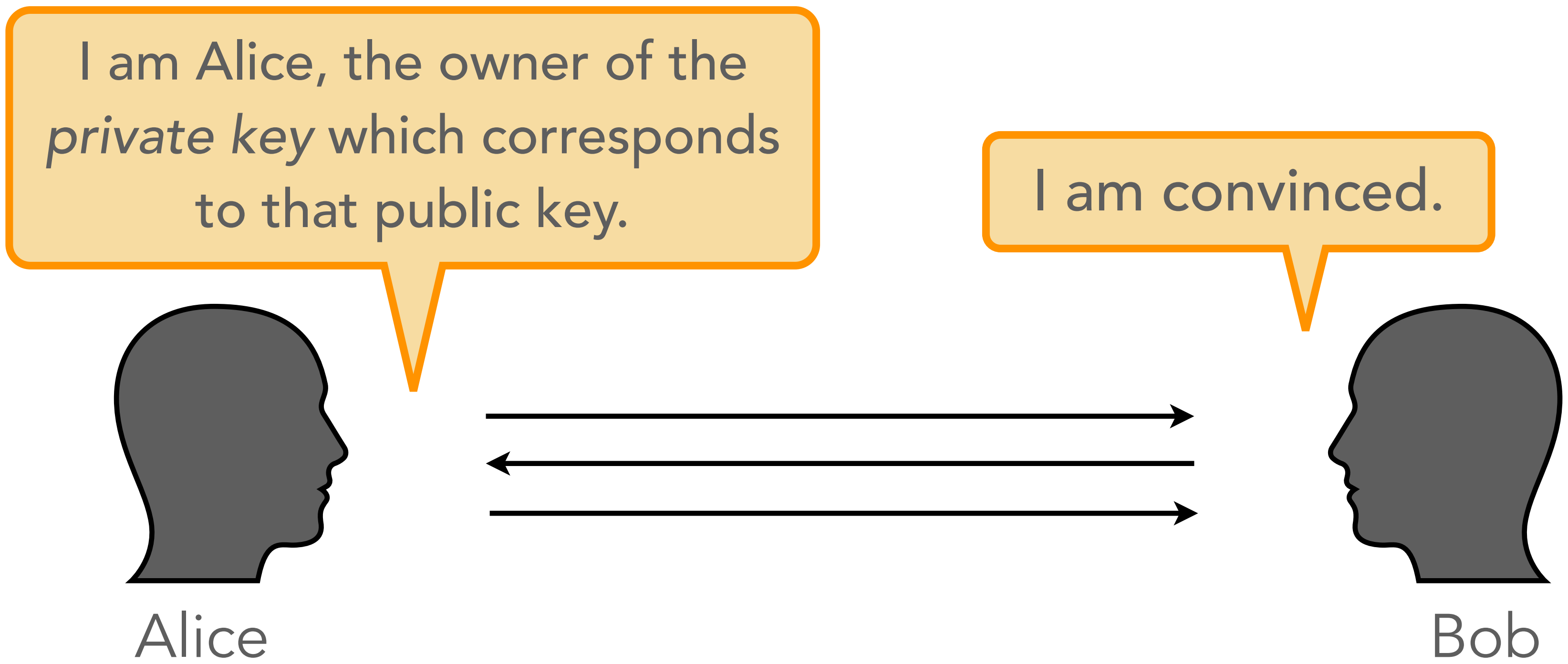


Authentication



Identification Scheme

Authentication



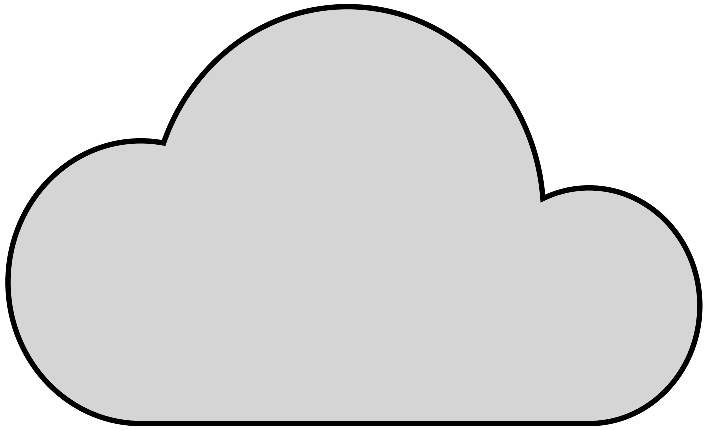
Identification Scheme

Can be transformed into (Fiat-Shamir transformation)

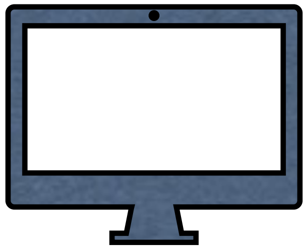
Signature Scheme

Computation Delegation

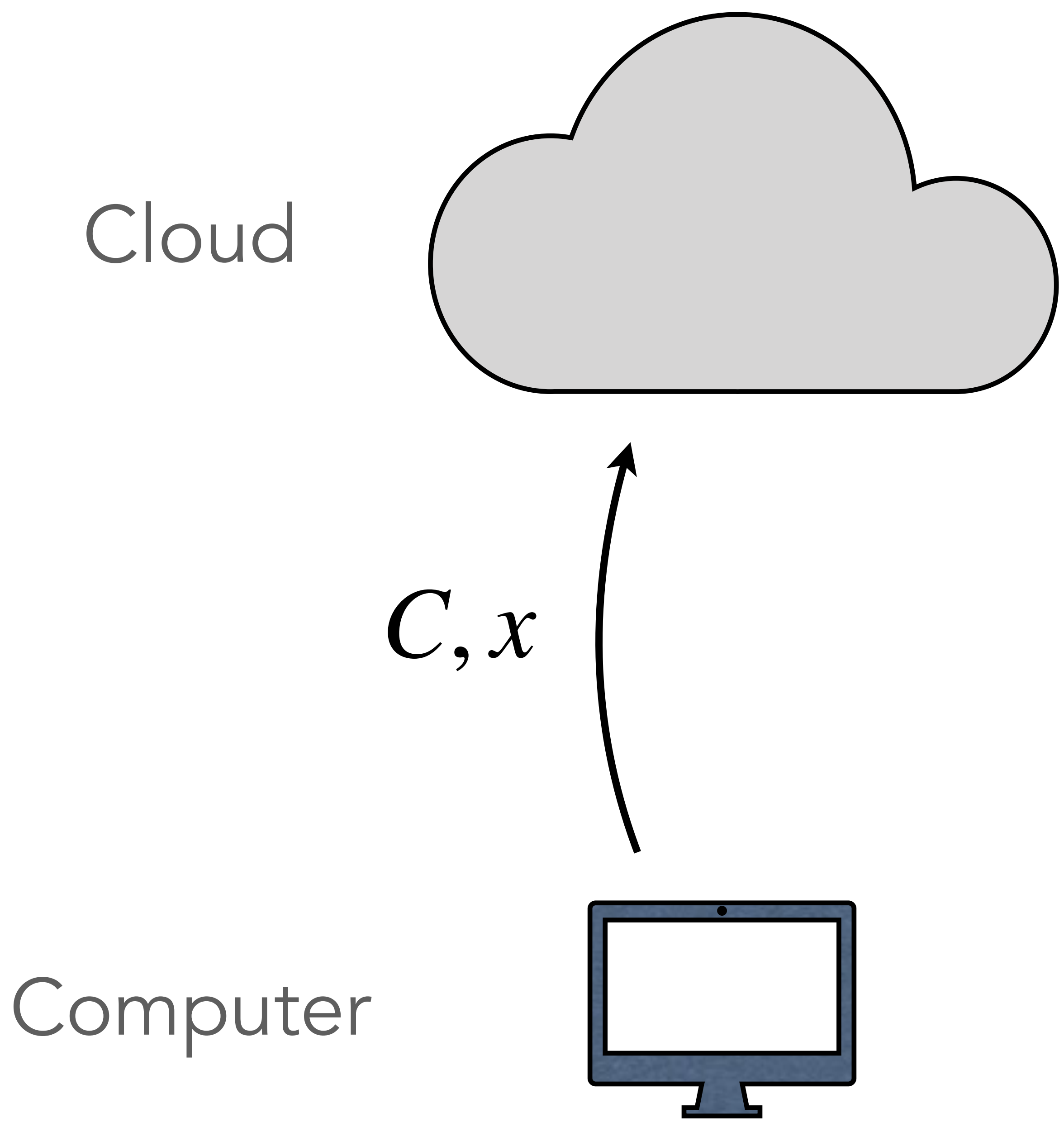
Cloud



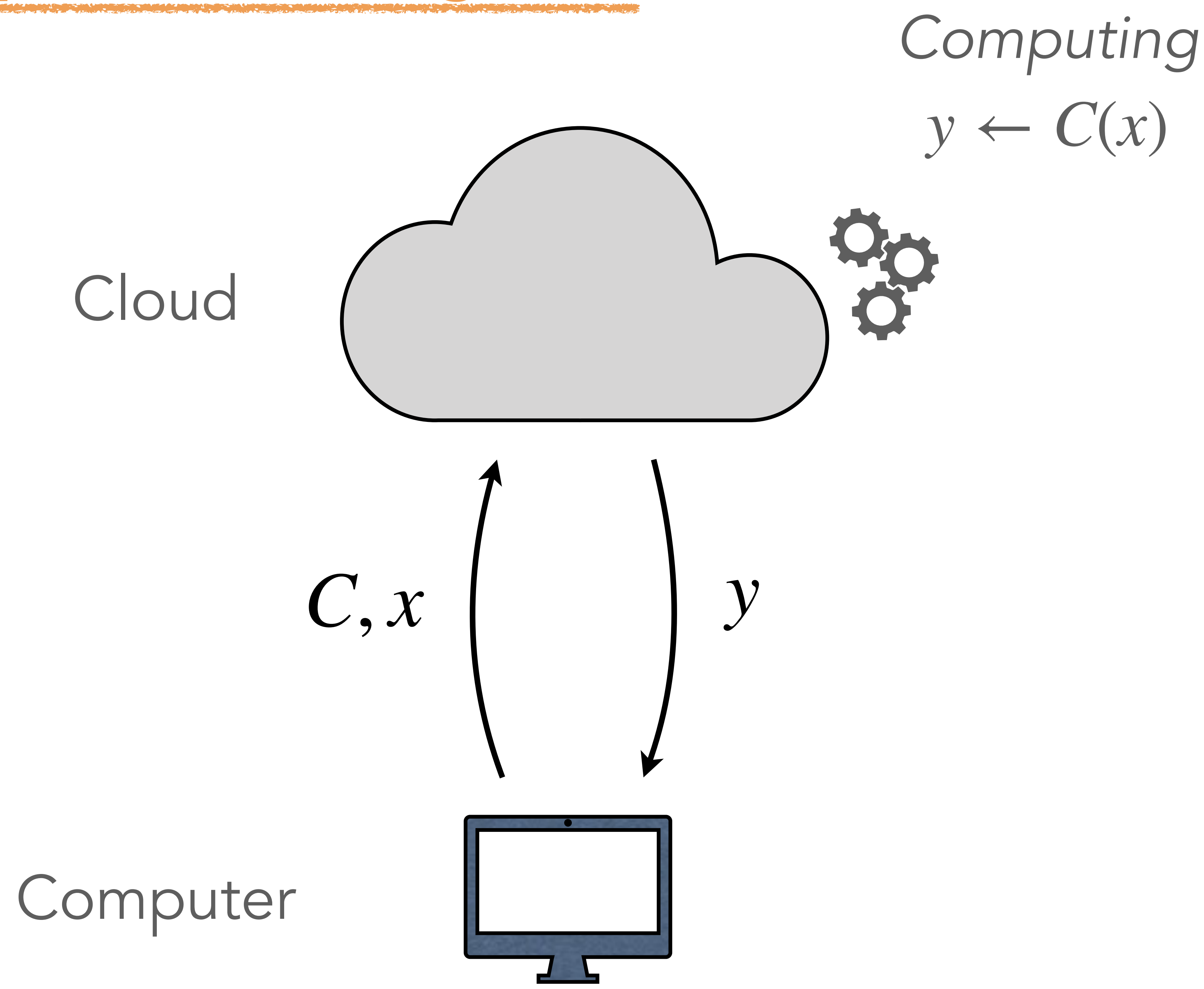
Computer



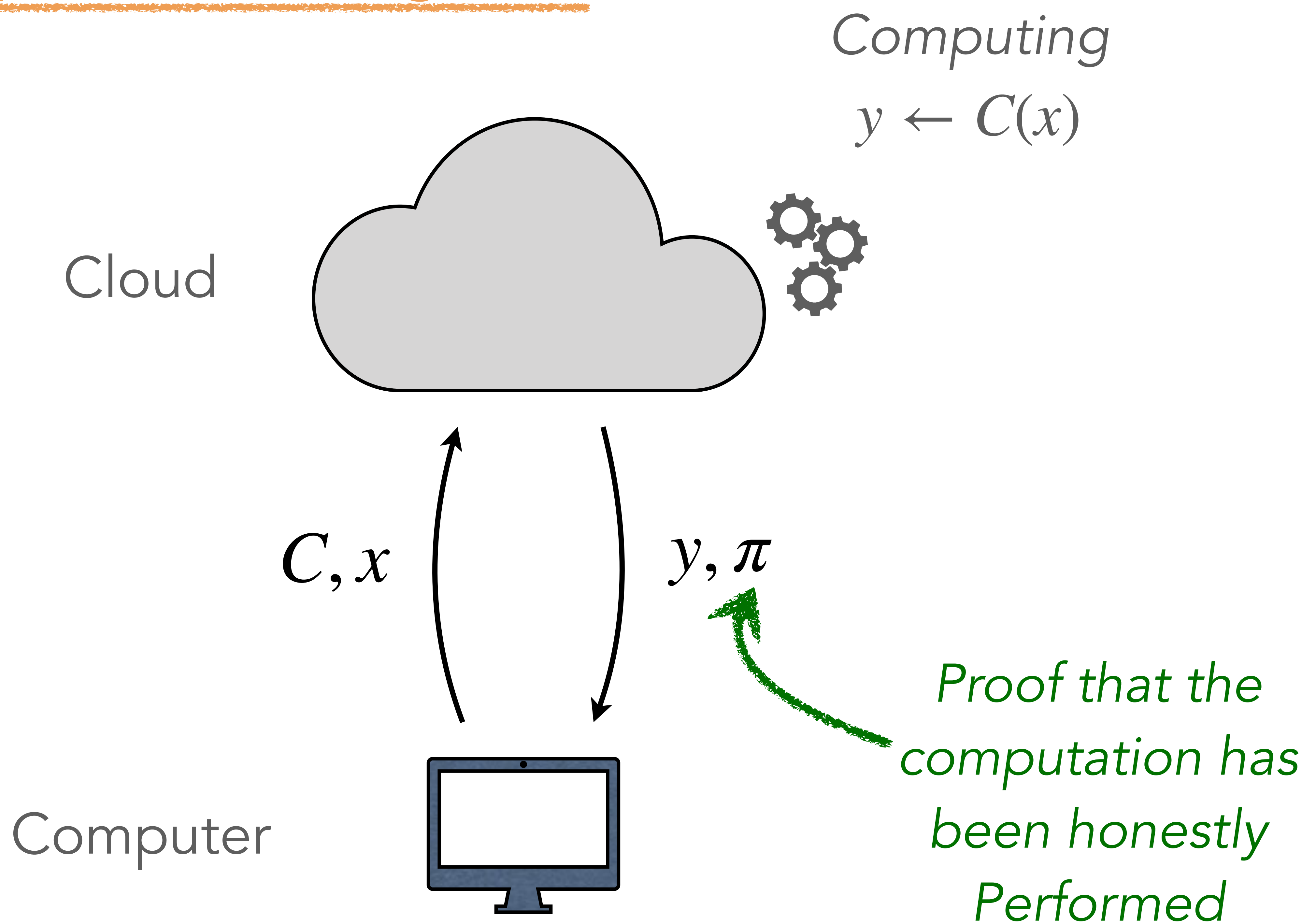
Computation Delegation



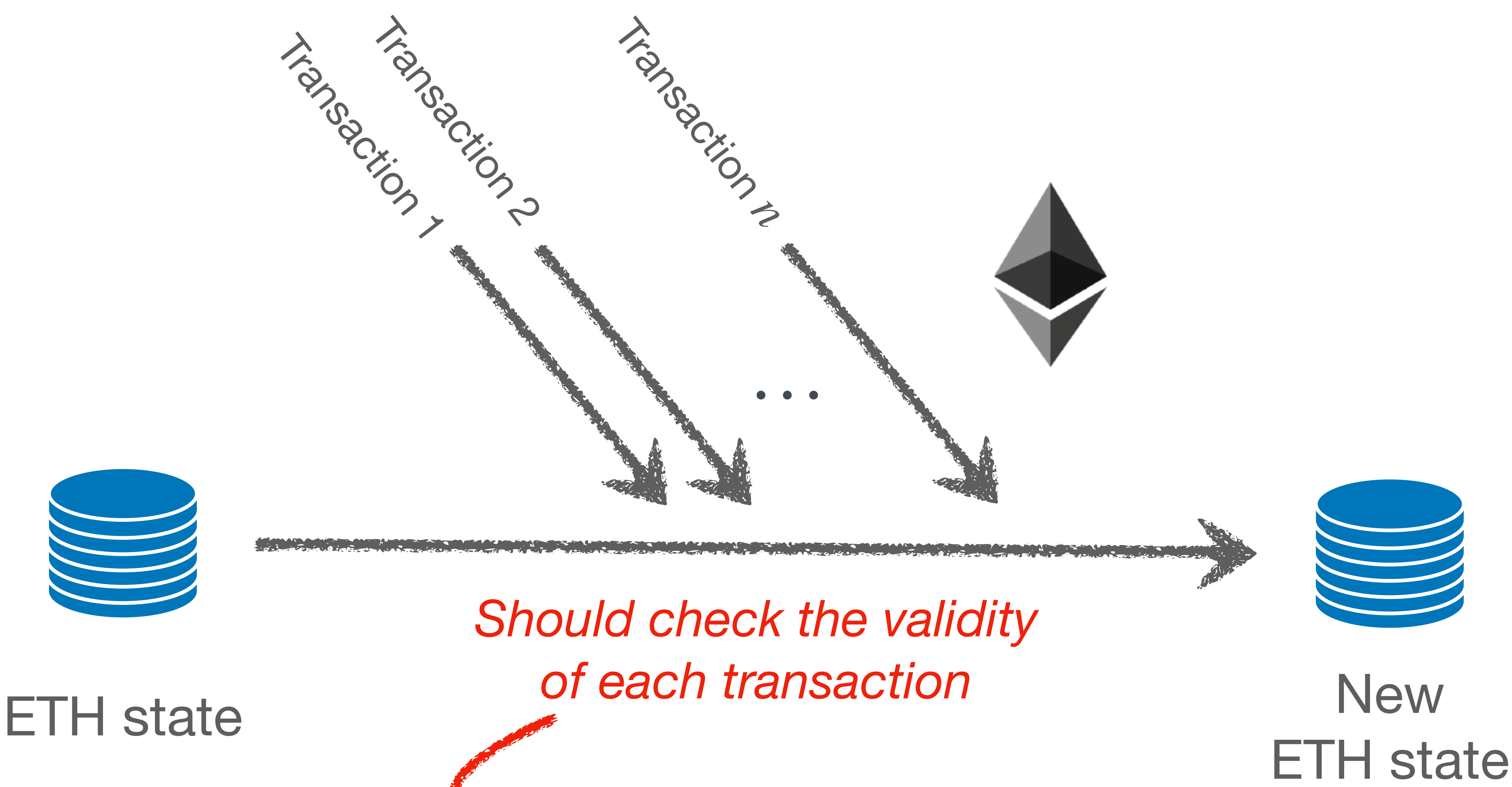
Computation Delegation



Computation Delegation



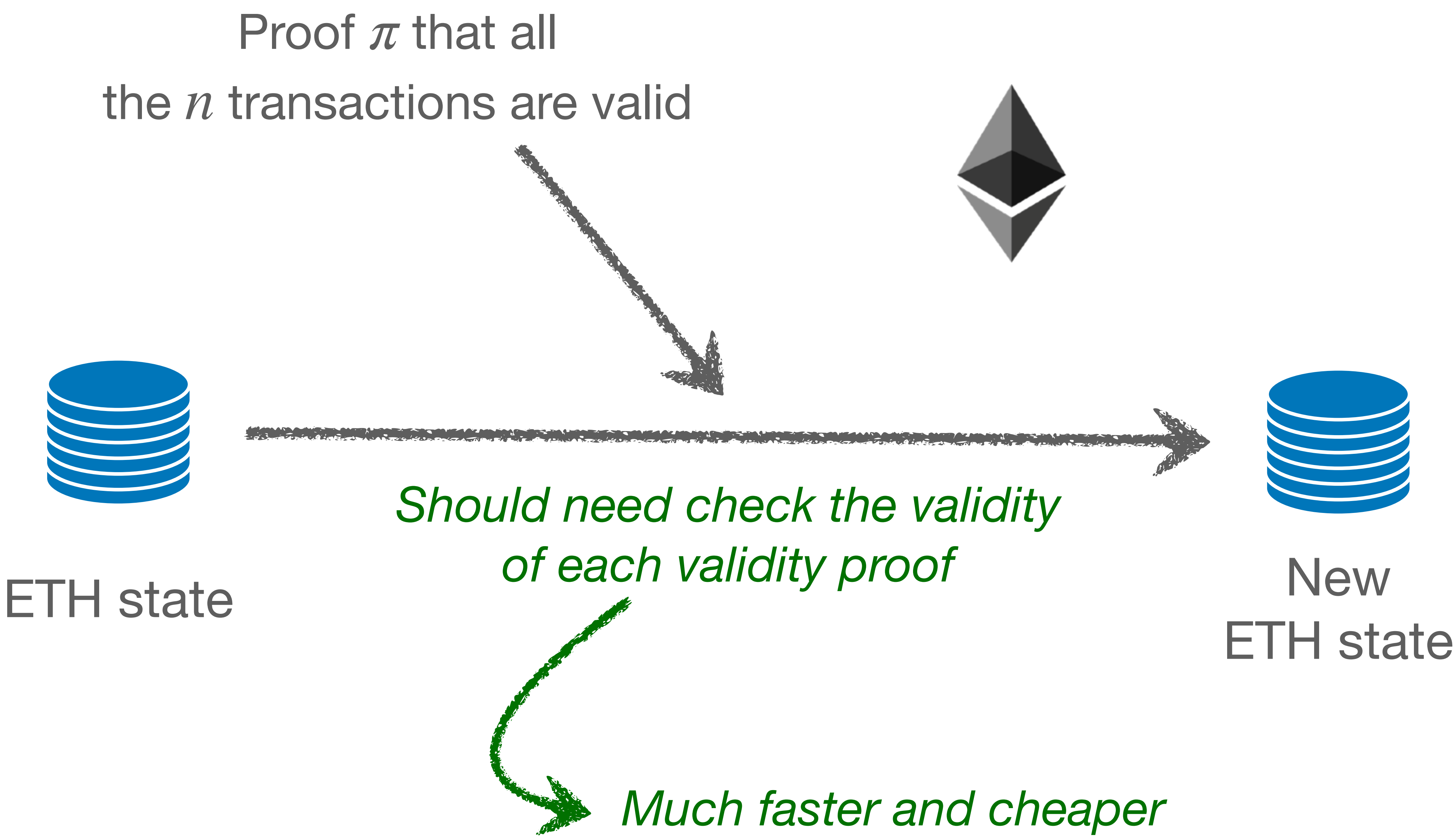
Computation Delegation (Blockchain)



It scales linearly into the number of transactions

$\Longrightarrow$ *It costs a lot of gaz.*

Computation Delegation (Blockchain)



Fighting Disinformation

How to verify where and when a photography was taken?

The Coalition for Content Provenance and Authenticity (C2PA) proposed a standard to verify image provenance that relies on digital signatures.

Cameras would "digitally sign" each photo taken along with a series of assertions about the photo (e.g., location, timestamp).

Fighting Disinformation

Cameras would "digitally sign" each photo taken along with a series of assertions about the photo (e.g., location, timestamp).

But in practice, the photos are often cropped, resized (etc.) before publishing. The photo signature would not be valid anymore.

When modifying a photo, we can produce a proof that the resulting photo corresponds to the original signed photos with a list of edits.

Conclusion

Conclusion

- Proof Systems:
  - Main properties:

    Completeness & Soundness

  - Additional optional properties:

    Zero-Knowledge, Efficient Verification,
    Short Communication, Non-interactive, Quantum-Resilient

- SNARK: Succinct Non-interactive Argument of Knowledge
  - Introduced in 2011
  - Different setups: trusted setup by circuit, universal trusted setup, transparent setup
  - SNARK = Functional Commitment Scheme + Interactive Oracle Proof

Conclusion

- Applications of proof systems:
  - Authentication (identification scheme)
  - Computation Delegation
    - Many use cases in blockchain
  - Disinformation Fight
  - E-voting
  - …

Thank you for your attention !